

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**ESPOL**

**MATERIA: PROGRAMACION DE SISTEMAS TELEMATICOS**

**TÍTULO DEL INFORME:**

**“DISPENSADOR DE COMIDA PARA MASCOTAS”**

**INTEGRANTES:**

**CALI LOAIZA BRYAN FABRICIO**

**TORRES CEDEÑO ROBERTO JOSUE**

**AREVALO GOYES JONNATHAN STEVEN**

**ALVARADO PELAEZ DANIEL ANDRES**

**GARCIA ROMO TAMARA KARINA**

**PARALELO: 1**

**PROFESOR:**

**MSIG. COLLAGUAZO JARAMILLO ADRIANA ELISA**

**FECHA DE PRESENTACIÓN:**

**MARTES, 1 DE SEPTIEMBRE DEL 2020**

**GUAYAQUIL - ECUADOR**

1. **Marco Teórico.**

**1.1. BASES TEÓRICAS**

**1.1.1. Bases teóricas veterinarias**

La alimentación de las mascotas es un tema muy importante que tratar, para una buena de la mascota. Así como el ser humano se llega a enfermar por una mala alimentación, nuestras mascotas no están exentas de ello, por lo cual pueden sufrir varias enfermedades como:

* Diarrea: Son cuando las heces se presentan en forma blanda o liquida, producto de alimentos descompuestos o por una variación en su dieta.
* Gastritis: Se produce por la ingesta de alimentos caducos o en malas condiciones, así también por alimentos caseros, que pueden hallar las mascotas en la basura.
* Desnutrición: Sucede cuando tienen una dieta desequilibrada por falta de nutrientes, por lo general suele presentarse en las mascotas sin hogar, pero también las de casa puede llegar a padecer.

**1.1.2. Bases teóricas electrónicas**

**1.1.2.1. Microcontrolador**

Es un circuito integrado utilizado en todo tipo de inventos que sea automático dependiendo de la necesidad del usuario; mejor comparado como una minicomputadora que controla elementos de entrada y salida; al tener procesador, memoria flash y RAM.

**1.1.2.2 Principio motor DC**

Basado en la ley de Fuerza, al recibir una corriente eléctrica un conductor es sumergido en un campo magnético, por lo cual recibe una fuerza perpendicular al plano formado por la corriente y el campo, según la regla de la mano derecha.

**1.1.3. Bases teóricas sistemas**

**1.1.3.1. Lenguaje C++**

El lenguaje C++ esta fundado por C, siendo uno de los lenguajes más antiguos, desarrollado desde 1979. Se tuvo que un tiempo fuera de la disponibilidad del público hasta 1985, siendo hasta la fecha un lenguaje muy popular; siendo un programa estándar ISO (Organización Internacional de Normalización) siendo desde bajo nivel y eficiente hasta ser complejo con optimas capacidades. Este lenguaje gana público por su biblioteca estándar, estando a la mano para empresas y terceros que desean crear su propio programa mediante este lenguaje.

**1.1.3.2. Lenguaje Java**

Es una plataforma y lenguaje informático comercializado desde 1995 por Sun Microsystems, siendo en la actualidad necesario su instalación para muchas aplicaciones y sitios web; su popularidad se debió a que es seguro, rápido y fiable. Siendo un lenguaje multiplataforma, estando en portátiles, consolas y teléfonos.

**1.2. MARCO CONCEPTUAL**

**1.2.1. Marco conceptual veterinario**

La alimentación de las mascotas depende del desgate físico y las calorías quemadas, para así determinar la cantidad de proteínas que faltan en su cuerpo; en la cual recuperara esa energía con comida especial para ellos y agua.

También es de tener muy en cuenta la edad de la mascota y su raza al saber si es pequeña, mediana o grande. Las vitaminas necesarias de una cría son mucho mayores que la de una mascota adulta.

**1.2.1.1. Tabla de alimentación de las mascotas según su tamaño y peso.**

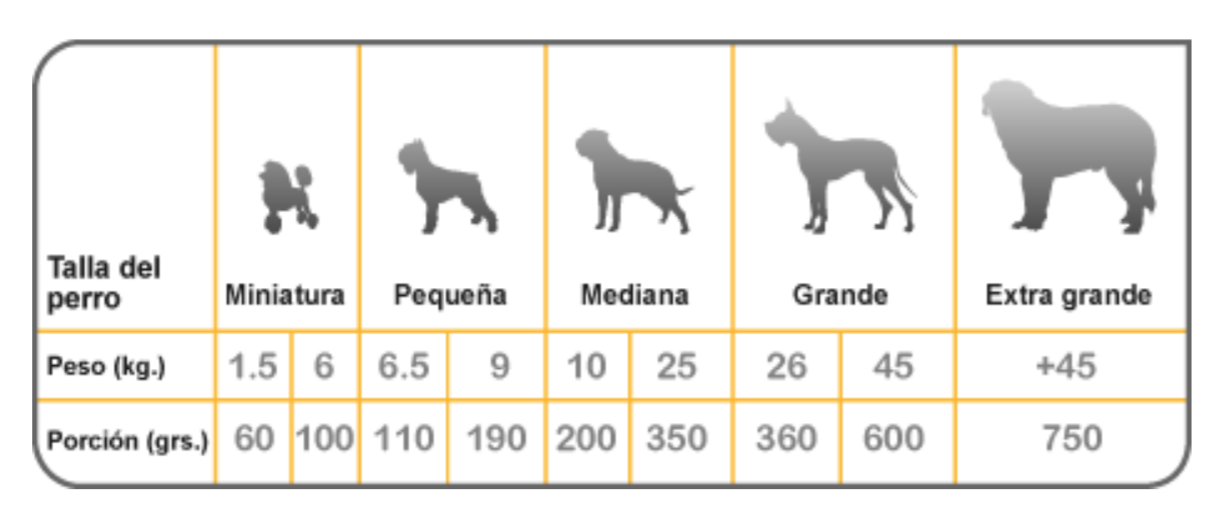


Figura 1 Porción de comida según el peso de la mascota

Fuente: (Nutrición Canis, 2010)

**1.2.2. Marco conceptual sistemas**

**1.2.2.1. Software Android**

Es un sistema operativo móvil para teléfonos inteligentes o tablets, que actualmente también lo encontramos en relojes inteligentes, televisores e incluso algunos modelos de carros. Fue desarrollado por Google y basado en Kernel de Linux, con códigos abiertos, para desarrolladores que desean crear diversas aplicaciones.

**1.2.3 Marco conceptual electrónico**

**1.2.3.1. Arduino UNO**

Es una placa muy utilizada al iniciar en el mundo de la programación y electrónica; siendo este una buena opción para tener una buena experiencia como principiantes, al tener mayor cantidad de información entre todos los demás tipos de Arduino.

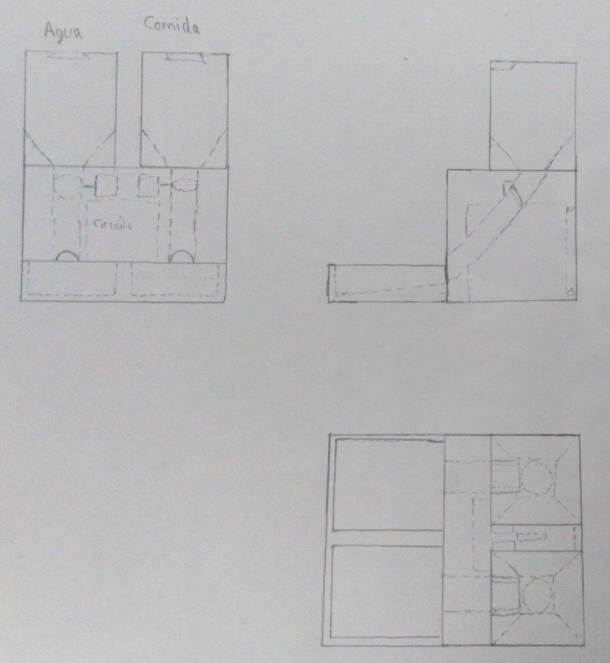
Este tipo de Arduino UNO, está basado en un microcontrolador ATmega328P. Este compuesto por 14 pines de entrada y salida digitales, con 6 entradas analógicas, una conexión USB, conector Jack de alimentación, y un botón de RESET. El Arduino UNO se lo debe conectar con un transformador AC-DC o por el puerto USB.

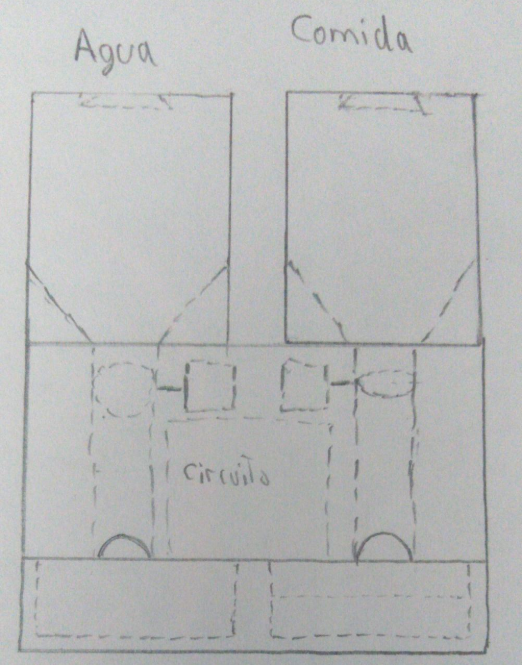
**1.2.3.2 Motor DC**

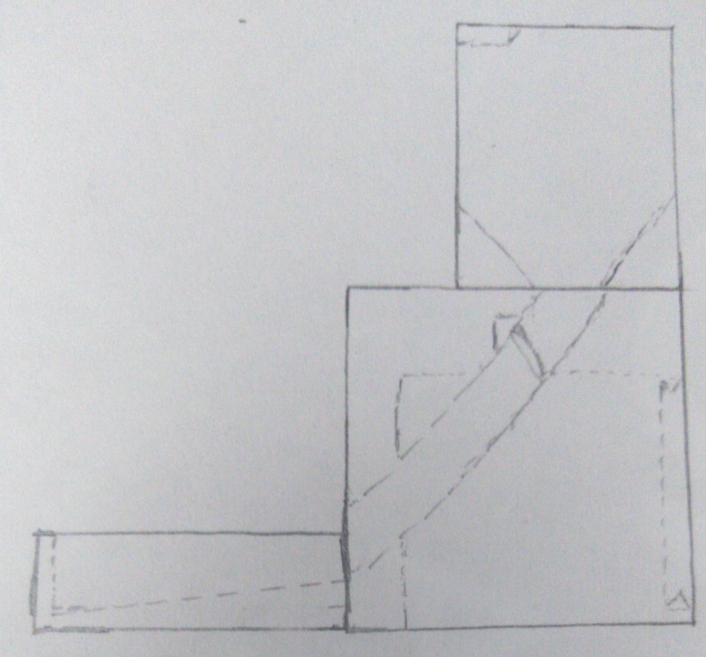
Es un motor de corriente continua, que su principal función es convertir la energía eléctrica en mecánica, al producir un movimiento rotatorio, generado por un campo magnético. Este tipo de motor está compuesto por dos partes:

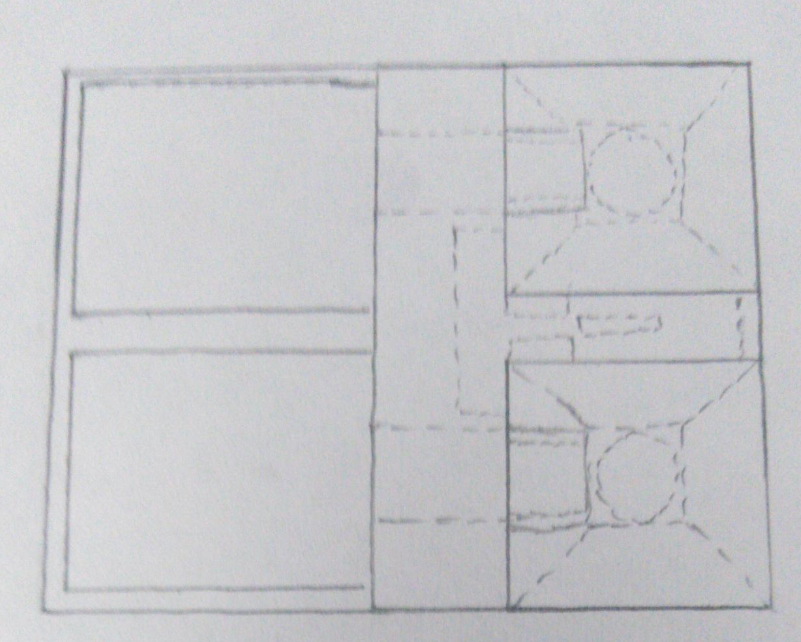
* Estator: Brinda soporte mecánico siendo la parte fija, conteniendo los polos siendo imanes permanentes o devanados.
* Rotor: LA mayoría de las veces lo encontramos en forma cilíndrica, con núcleo y devanado, nutrido con corriente a través de escobillas fijas de carbono.

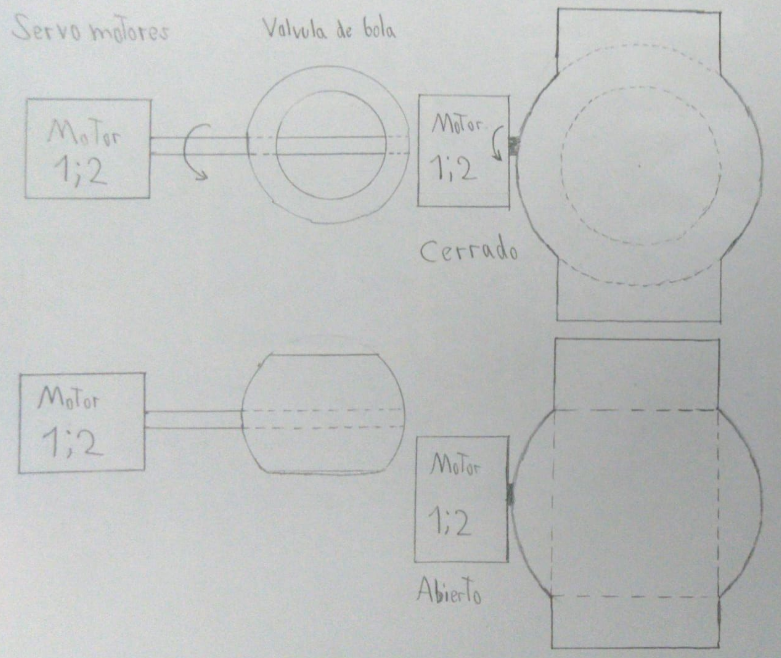
1. **Diagrama del diseño.**

****

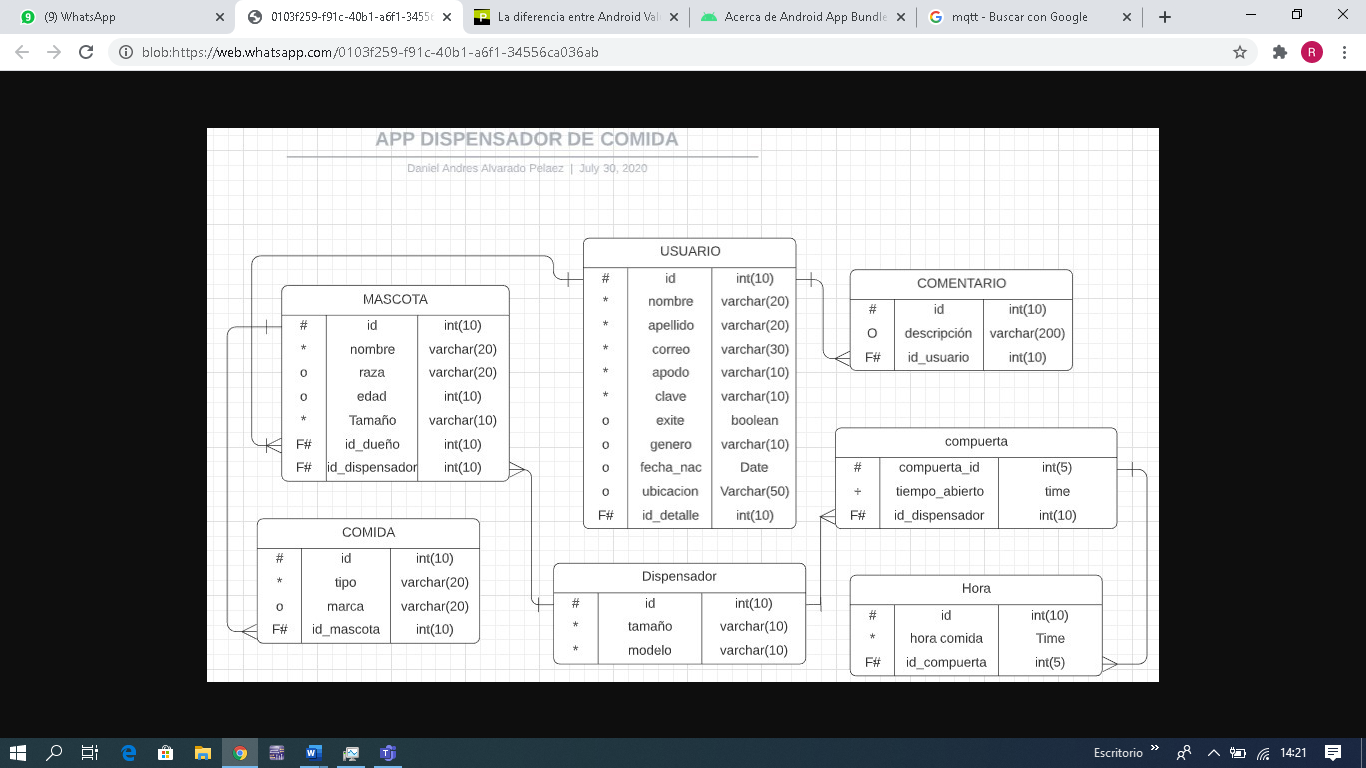
****

****



****

1. **Modelo entidad-relación.**



1. **Diagrama de circuito.**

Se utilizará un modulo de red Ethernet Shield w5100, el cual es montado en el arduino de la siguiente manera (se utilizará un arduino mega, pero el montaje no cambia).

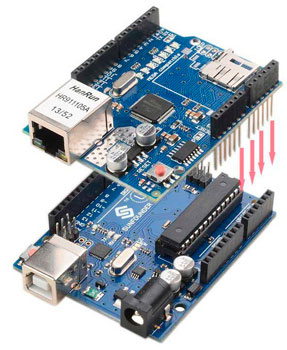
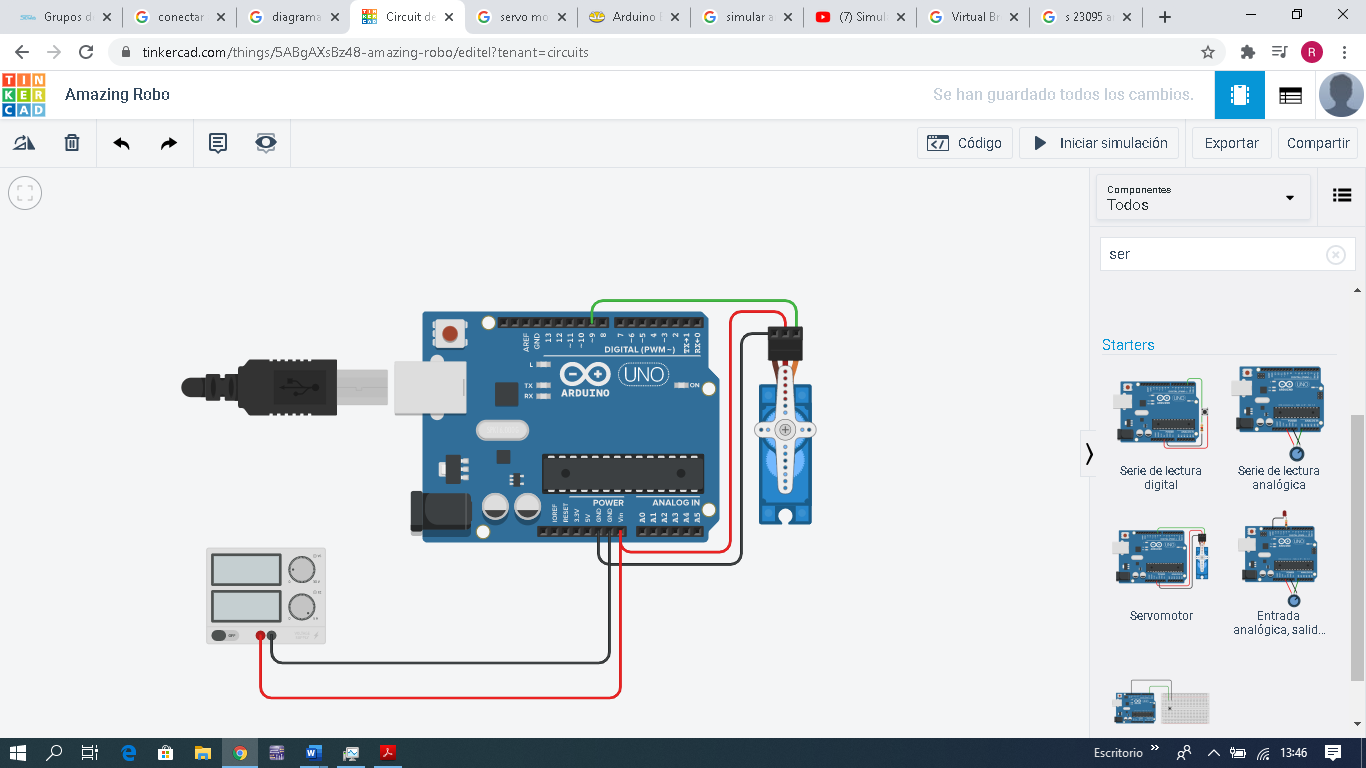


Diagrama para el servomotor, conectado a una fuente de alimentación externa de 10 volteos (se utilizará un arduino mega, pero las conexiones no varían).



1. **Recursos de hardware.**

Los recursos utilizados para el hardware serán:

* Arduino mega.
* Módulo de red Arduino Ethernet Shield w5100.
* Celular con sistema operativo Android.
* Servomotor MG995.
* Fuente de alimentación externa de 10 volteos para el arduino
* Diseño externo del prototipo.
* Es obligatorio tener acceso a conexión Ethernet

1. **Recursos de software.**

Para el diseño y la programación de la aplicación móvil utilizaremos el software Android Studio, para la progresión del Arduino utilizaremos el software propio de él.

Se trabajará bajo el sistema operativo de Android, el lenguaje de programación de Android Studio es Java y para el Arduino es C++.

Para hacer la conexión al puerto Ethernet por medio del Arduino Ethernet Shield, se importará la librería #include <Ethernet.h> y #include <SPI.h>, por parte de Android Studio, las librerías de conectividad son import org.apache.http.client.HttpClient;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.impl.client.DefaultHttpClient;

1. **Preguntas técnicas del informe.**

**¿Cómo funciona la solución propuesta?**

Funciona de la siguiente manera: el usuario del dispensador de comida tendrá que descargar una aplicación móvil desde Play Store, creará un usuario y contraseña, el usuario tendrá que contestar unas preguntas acerca de su mascota (promedio en que hace cambio de agua y comida a la mascota, raza, tamaño, etc.) y preguntas sobre cómo se puede mejorar la aplicación y el sistema de alimentación, para posterior generar una base de datos con esta información.

En la aplicación existirá botones manuales que podrá controlar el usuario para llenar el plato de croqueta o el plato del agua cuando se percate que los platos están vacíos o la mascota pide comida o agua, así mismo el sistema interno tendrá definido por medio de la base de datos, horas promedios en que automáticamente el alimentador puede arrojar comida o agua a los platos respectivamente (por lo regular se define como: 3 veces al día la comida y 1 vez al día el agua).

Se está pensado aun si contará la aplicación con una alarma, la cual indique al usuario que es el tiempo de darle de comer o darle agua a su mascota y si desea que se proceda a alimentarlo o a hidratarlo, así se evitara que el dispensador se active si el usuario ve que en ese tiempo aun existen comida en el plato. Evitando también que se desperdicie comida y ahorrando al cliente tener que ir hasta el lugar en donde le da comer al perro y que pueda seguir haciendo sus quehaceres.

**¿Qué van a construir a nivel de hardware para resolver el problema?**

A nivel de hardware se desea construir un alimentador que lleve la comida y el agua a través de dos tubos PVC, los cuales tendrán una válvula conectada a un servomotor (cada tubo), el cual será manejado por la lógica de la programación del arduino y de la aplicación respectivamente, se construirá una caja de madera para que ahí adentro este todo el circuito protegido y que los tubos no den una mala estética al hogar.

Se tendrá un módulo Arduino Ethernet que estará montada sobre el arduino mega y los servomotores que también estarán conectados al Arduino mega.

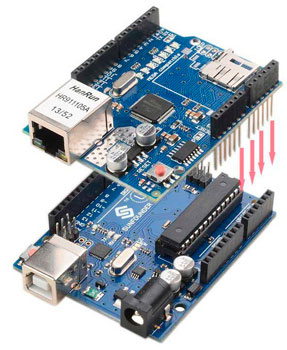
Por ultimo se necesitará obligatoriamente tener conexión ethernet en la casa en la que se vaya a montar el alimentador, un celular inteligente con sistema operativo Android y un enchufe para conectar el alimentador.

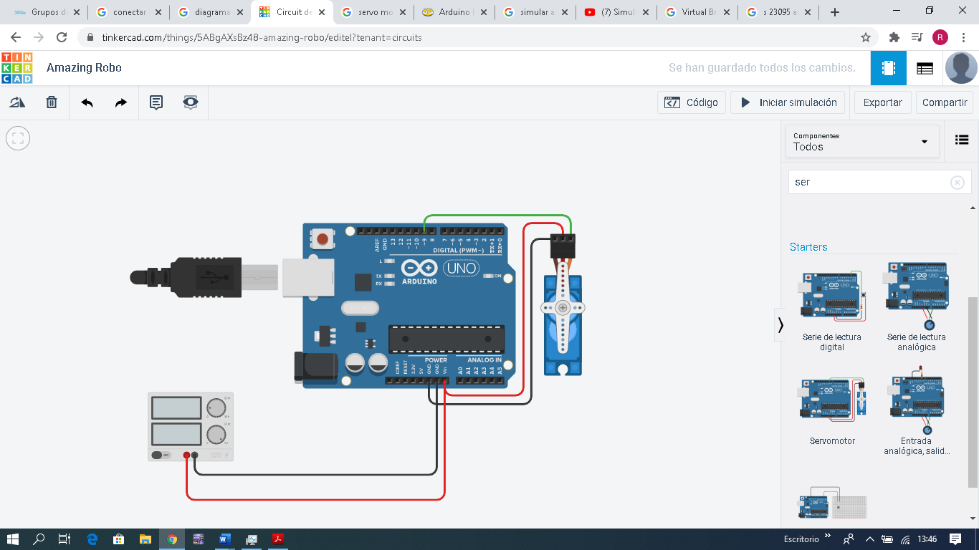
**¿Cómo este producto aporta al campo de investigación aplicada?**

El dispensador de comida tiene como objetivo usar conocimientos de programación de sistemas telemáticos, electrónica, diseño mecánico, física, entre otras áreas. Para crear un dispositivo que conectado a una red Ethernet sea capaz de ser programado según los requerimientos de o de los animales, que el usuario ingrese al sistema, además de automáticamente reconocer si es necesario el dispensar comida fuera de los horarios programados y notificar si se necesitan de un reabastecimiento de los alimentos.

Siendo así que consolidando los conocimientos de las áreas ya mencionadas se aplicará en la creación de una máquina que cuidará de la correcta alimentación de las mascotas del usuario. Y alertara de ser necesario de posibles sugerencias para mejorar los programas alimentarios.

1. **DIAGRAMA DEL PROYECTO**





1. **Base de datos**

La Base de datos de nuestra aplicación móvil es interna al programa, esta posee diferentes factores dentro de la misma, tanto del usuario como de la mascota, estos diferentes campos se dividen entre:  
-Nombre de Usuario: Consiste en la *primary key* de nuestra base de datos por lo que permitirá la existencia única para cada usuario que desee ingresar a nuestro sistema e implementar la aplicación.

-Nombre: Se trata del nombre del usuario o aquel que vaya a ingresar al sistema, este dato será necesario para poder referirse al usuario de manera directa y conocer dicho aspecto, importante ante la posibilidad de que exista alguna clase de reclamo o sugerencia sobre nuestro producto.

-Contraseña: Este campo es uno de los más importantes, con respecto a la seguridad que nuestra aplicación pueda implementar, debido a que cada usuario podrá acceder a su cuenta únicamente si ingresa correctamente su contraseña, de este modo en caso de que algún ente externo intente sustraer la información del usuario se le sea impedido dicho acto por medio de la contraseña.

-Ubicación: Un dato de igual importancia que el nombre ya que en caso de existir alguna petición mantenimiento de nuestro dispensador de alimentos para mascotas, usar dicho campo para poder ubicar la localización del usuario.

-Teléfono: Este campo será utilizado en caso de requerir contactar con el usuario por cualquier posible caso que pueda suscitar, por lo que hacer que el usuario llene este campo también es de suma relevancia.

En el caso de que el usuario genere alguna clase de notificación este también tendrá que llenar ciertos campos para que esto se realice, dichos campos también formaran parte de nuestra base de datos, y cada usuario puede generar uno o más horarios para las notificaciones.

-Horario: Este campo será llenado con un tipo data, por lo que implementamos en nuestro programa un método que le permite al usuario tener un reloj del cual escoger la hora, de esta manera se evita que los valores sean estrictamente de hora, y no existan fallos en el sistema en el caso de introducir algún Int o String.

-Descripción: Un campo opcional el cual puede, o no ser llenado por el usuario, este le permitirá dar algún mensaje propio a la notificación una vez que esta sea activada.

Para la introducción de los datos de la mascota, el usuario también dispondrá de espacio en nuestra base de datos para generar la información del mismo, este campo al igual que las notificaciones, pueden ser generadas de uno o más por cada usuario, y dicha información comprenderá el propósito principal de nuestro dispensador por lo que su inclusión es vital para nuestra aplicación.

-Nombre de Mascota: Este dato le permitirá conocer al usuario, en caso de que posea más de una mascota, una especificación detallada de entre las mascotas que posea para que no existan confusiones a la hora de alimentarlas, evitando que así alguna de ellas se quede sin comer.

-Raza: Un campo opcional, pero importante ya que le permitirá conocer el tipo de dieta que mejor se adecue para su mascota, y al igual que con el punto anterior, en caso de poseer más de una mascota, no existan fallos frente a su alimentación.

-Tamaño: Este punto es muy importante ya que denotara la funcionalidad de nuestra aplicación, ya que nuestro motor proporcionara alimento según sea el rasgo definido en este punto, siendo así que las porciones de comida serán de mayor o menor cantidad según el tamaño de la mascota.

A continuación, se mostrarán los códigos utilizados para generar nuestra base de datos dentro de la aplicación:

1. **Explicación del código**

**Paso: Creación de la base de datos**

Se creó un paquete denominado Utilidades y dentro de este, la clase Utilidades, en la cual se crearon las tablas que van a pertenecer a la base de datos.

**package** com.example.proyecto.utilidades;

**public class** Utilidades {  
  
 *//NOMBRES DE LAS TABLAS* **public static final** String ***TABLA\_USUARIO***= **"usuario"**;  
 **public static final** String ***TABLA\_MASCOTA***= **"mascota"**;  
 **public static final** String ***TABLA\_HORARIO***= **"horario"**;  
  
 *//COLUMNAS DE LA TABLA DE USUARIO* **public static final** String ***KEY\_USUARIO\_ID***= **"id"**;  
 **public static final** String ***KEY\_USUARIO\_NOMBREUSUARIO***= **"nombreUsuario"**;  
 **public static final** String ***KEY\_USUARIO\_NOMBRES***= **"nombres"**;  
 **public static final** String ***KEY\_USUARIO\_CONTRASEÑA***= **"contraseña"**;  
 **public static final** String ***KEY\_USUARIO\_CORREO***= **"correo"**;  
 **public static final** String ***KEY\_USUARIO\_UBICACION***=**"ubicacion"**;  
 **public static final** String ***KEY\_USUARIO\_TELEFONO***=**"telefono"**;  
  
  
 *//COLUMNAS DE LA TABLA DE MASCOTA* **public static final** String ***KEY\_MASCOTA\_ID***= **"id"**;  
 **public static final** String ***KEY\_MASCOTA\_ID\_USUARIO***= **"usuarioId"**;*//llave foranea* **public static final** String ***KEY\_MASCOTA\_NOMBRE***= **"mascotaNombre"**;  
 **public static final** String ***KEY\_RAZA***= **"raza"**;  
 **public static final** String ***KEY\_MASCOTA\_EDAD***= **"edadMascota"**;  
 **public static final** String ***KEY\_TAMAÑO***=**"tamaño"**;  
  
 *//COLUMNAS DE LA TABLA DE HORARIO* **public static final** String ***KEY\_HORARIO\_ID***= **"id"**;  
 **public static final** String ***KEY\_HORARIO\_HORA***= **"hora"**;  
 **public static final** String ***KEY\_HORARIO\_DESCRIPCION***= **"descripcion"**;  
 **public static final** String ***KEY\_HORARIO\_IDUSUARIO***= **"idUsuario"**;  
 **public static final** String ***KEY\_HORARIO\_TONO***=**"tono"**;  
  
  
 *//CREACION DE LAS TABLAS* **public static final** String ***CREAR\_TABLA\_USUARIO***=**"CREATE TABLE "** +  
 **""**+***TABLA\_USUARIO***+**" ("**+***KEY\_USUARIO\_ID*** + **" "**+  
 **"INTEGER PRIMARY KEY AUTOINCREMENT,"**+  
 ***KEY\_USUARIO\_NOMBREUSUARIO*** + **" VARCHAR,"** +  
 ***KEY\_USUARIO\_NOMBRES*** + **" VARCHAR,"** +  
 ***KEY\_USUARIO\_CONTRASEÑA*** + **" VARCHAR,"** +  
 ***KEY\_USUARIO\_CORREO*** + **" VARCHAR,"** +  
 ***KEY\_USUARIO\_UBICACION***+ **" VARCHAR,"**+  
 ***KEY\_USUARIO\_TELEFONO***+ **" VARCHAR)"**;  
  
 **public static final** String ***CREAR\_TABLA\_MASCOTA***=**"CREATE TABLE "** +  
 **""**+***TABLA\_MASCOTA***+**" ("**+***KEY\_MASCOTA\_ID*** + **" INTEGER PRIMARY KEY AUTOINCREMENT,"**+  
 ***KEY\_MASCOTA\_ID\_USUARIO***+ **" INTEGER REFERENCES "**+ ***TABLA\_USUARIO***+**","**+  
 ***KEY\_MASCOTA\_NOMBRE*** + **" VARCHAR,"** +  
 ***KEY\_RAZA*** + **" VARCHAR,"** +  
 ***KEY\_MASCOTA\_EDAD*** + **" VARCHAR,"** +  
 ***KEY\_TAMAÑO***+ **" VARCHAR)"**;  
  
 **public static final** String ***CREAR\_TABLA\_HORARIO***=**"CREATE TABLE "** +  
 **""**+***TABLA\_HORARIO***+**" ("**+***KEY\_HORARIO\_ID*** + **" "**+  
 **"INTEGER PRIMARY KEY AUTOINCREMENT,"**+***KEY\_HORARIO\_HORA***+ **" VARCHAR,"**+  
 ***KEY\_HORARIO\_DESCRIPCION*** + **" VARCHAR,"** +  
 ***KEY\_HORARIO\_IDUSUARIO***+ **" INTEGER REFERENCES "**+ ***TABLA\_USUARIO***+**","**+  
 ***KEY\_HORARIO\_TONO***+ **" VARCHAR)"**;}

**AdminSQLiteOpenHelper.java**

Para crear la base de datos se creó la clase *AdminSQLiteOpenHelper* que extiende *SQLiteOpenHelper* la cual nos permitio trabajar de forma local en nuestros dispositivos reales y se encuentra dentro del paquete entidades.

**package** com.example.proyecto.entidades;  
  
**import** android.content.Context;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteOpenHelper;  
  
**import** com.example.proyecto.utilidades.\*;

**public class** AdminSQLiteOpenHelper **extends** SQLiteOpenHelper {  
  
 **public** AdminSQLiteOpenHelper(Context context,String name, SQLiteDatabase.CursorFactory factory, **int** version) {  
 **super**(context, name, factory, version);  
 }  
  
  
 *//LAS TABLAS SE CREARON DENTRO DEL PAQUETE UTILIDADES* @Override  
 **public void** onCreate(SQLiteDatabase db) {  
 db.execSQL(Utilidades.***CREAR\_TABLA\_MASCOTA***);  
 db.execSQL(Utilidades.***CREAR\_TABLA\_USUARIO***);  
 db.execSQL(Utilidades.***CREAR\_TABLA\_HORARIO***);  
  
 }  
  
 @Override  
 **public void** onUpgrade(SQLiteDatabase db, **int** versionAntigua, **int** versionNueva) {  
 db.execSQL(**"DROP TABLE IF EXISTS "**+Utilidades.***TABLA\_USUARIO***);  
 db.execSQL(**"DROP TABLE IF EXISTS "**+Utilidades.***TABLA\_MASCOTA***);  
 db.execSQL(**"DROP TABLE IF EXISTS "**+Utilidades.***TABLA\_HORARIO***);  
  
 onCreate(db);  
 }  
}

**Paso: Entidades**

Se crearon tres entidades principales para manejar nuestra base de datos y relacionar un usuario con sus mascotas y alarmas de notificación, dichas entidades se implementan de Serializable.

**Usuario.java**

Esta clase contiene los parámetros necesarios para un usuario y sus respectivos métodos y constructores para poder acceder a su información.

**package** com.example.proyecto.entidades;  
**import** java.io.Serializable;  
  
**public class** Usuario **implements** Serializable {  
  
 **private** Integer **id**;  
 **private** String **nombreUsuario**;  
 **private** String **nombres**;  
 **private** String **contraseña**;  
 **private** String **correo**;  
 **private** String **ubicacion**;  
 **private** String **telefono**;  
  
 **public** Usuario() {  
 }  
  
 **public** Usuario(Integer id, String nombreUsuario, String nombres, String contraseña, String correo, String ubicacion, String telefono) {  
 **this**.**id** = id;  
 **this**.**nombreUsuario** = nombreUsuario;  
 **this**.**nombres** = nombres;  
 **this**.**contraseña** = contraseña;  
 **this**.**correo** = correo;  
 **this**.**ubicacion** = ubicacion;  
 **this**.**telefono** = telefono;  
 }  
  
 **public** Integer getId() {  
 **return id**;  
 }  
  
 **public void** setId(Integer id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getNombreUsuario() {  
 **return nombreUsuario**;  
 }  
  
 **public void** setNombreUsuario(String nombreUsuario) {  
 **this**.**nombreUsuario** = nombreUsuario;  
 }  
  
 **public** String getNombres() {  
 **return nombres**;  
 }  
  
 **public void** setNombres(String nombres) {  
 **this**.**nombres** = nombres;  
 }  
  
 **public** String getContraseña() {  
 **return contraseña**;  
 }  
  
 **public void** setContraseña(String contraseña) {  
 **this**.**contraseña** = contraseña;  
 }  
  
 **public** String getCorreo() {  
 **return correo**;  
 }  
  
 **public void** setCorreo(String correo) {  
 **this**.**correo** = correo;  
 }  
  
 **public** String getUbicacion() {  
 **return ubicacion**;  
 }  
  
 **public void** setUbicacion(String ubicacion) {  
 **this**.**ubicacion** = ubicacion;  
 }  
  
 **public** String getTelefono() {  
 **return telefono**;  
 }  
  
 **public void** setTelefono(String telefono) {  
 **this**.**telefono** = telefono;  
 }  
}

**Mascota.java**

Contiene los parámetros necesarios de una mascota y sus respectivos constructores y métodos.

**package** com.example.proyecto.entidades;  
  
**import** java.io.Serializable;

**public class** Mascota **implements** Serializable {  
  
 **public** Integer **idMascota**;  
 **public** Usuario **dueño**;  
 **public** Integer **idUsuario**;  
 **public** String **nombreMascota**;  
 **public** String **raza**;  
 **public** Integer **edad**;  
 **public** String **tamaño**;  
  
 **public** Mascota() {  
 }  
  
 **public** Mascota(Integer idMascota, Integer idUsuario, String nombreMascota, String raza, Integer edad, String tamaño) {  
 **this**.**idUsuario** = idUsuario;  
 **this**.**idMascota** = idMascota;  
 **this**.**nombreMascota** = nombreMascota;  
 **this**.**raza** = raza;  
 **this**.**edad** = edad;  
 **this**.**tamaño** = tamaño;  
 }  
  
  
 **public** Mascota(Integer idMascota, Usuario dueño, Integer idUsuario, String nombreMascota, String raza, Integer edad, String tamaño) {  
 **this**.**idMascota** = idMascota;  
 **this**.**dueño** = dueño;  
 **this**.**idUsuario** = idUsuario;  
 **this**.**nombreMascota** = nombreMascota;  
 **this**.**raza** = raza;  
 **this**.**edad** = edad;  
 **this**.**tamaño** = tamaño;  
 }  
  
 **public** Integer getIdUsuario() {  
 **return idUsuario**;  
 }  
  
 **public void** setIdUsuario(Integer idUsuario) {  
 **this**.**idUsuario** = idUsuario;  
 }  
  
 **public** Integer getIdMascota() {  
 **return idMascota**;  
 }  
  
 **public void** setIdMascota(Integer idMascota) {  
 **this**.**idMascota** = idMascota;  
 }  
  
 **public** String getNombreMascota() {  
 **return nombreMascota**;  
 }  
  
 **public void** setNombreMascota(String nombreMascota) {  
 **this**.**nombreMascota** = nombreMascota;  
 }  
  
 **public** String getRaza() {  
 **return raza**;  
 }  
  
 **public void** setRaza(String raza) {  
 **this**.**raza** = raza;  
 }  
  
 **public** Integer getEdad() {  
 **return edad**;  
 }  
  
 **public void** setEdad(Integer edad) {  
 **this**.**edad** = edad;  
 }  
  
 **public** String getTamaño() {  
 **return tamaño**;  
 }  
  
 **public void** setTamaño(String tamaño) {  
 **this**.**tamaño** = tamaño;  
 }  
 }

**Horario.java**

**package** com.example.proyecto.entidades;  
  
**import** android.media.ToneGenerator;  
**import** java.io.Serializable;  
  
**public class** Horario **implements** Serializable {  
 **private** Integer **id**;  
 **private** String **hora**;   
 **private** String **descripcion**;  
 **private** Integer **iDUsuario**;  
 **private** ToneGenerator **Tono**;  
  
 **public** Horario(){  
 }  
  
 **public** Horario(Integer id, String hora, String descripcion, Integer iDUsuario, ToneGenerator tono) {  
 **this**.**id** = id;  
 **this**.**hora** = hora;  
 **this**.**descripcion** = descripcion;  
 **this**.**iDUsuario** = iDUsuario;  
 **Tono** = tono;  
 }  
 **public** Integer getId() {  
 **return id**;  
 }  
  
 **public void** setId(Integer id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getHora() {  
 **return hora**;  
 }  
  
 **public void** setHora(String hora) {  
 **this**.**hora** = hora;  
 }  
  
 **public** String getDescripcion() {  
 **return descripcion**;  
 }  
  
 **public void** setDescripcion(String descripcion) {  
 **this**.**descripcion** = descripcion;  
 }  
  
 **public** Integer getiDUsuario() {  
 **return iDUsuario**;  
 }  
  
 **public void** setiDUsuario(Integer iDUsuario) {  
 **this**.**iDUsuario** = iDUsuario;  
 }  
  
 **public** ToneGenerator getTono() {  
 **return Tono**;  
 }  
  
 **public void** setTono(ToneGenerator tono) {  
 **Tono** = tono;  
 }}

**Paso: Registro e inicio de sesión del Usuario**

**RegistroUsuario.java**

En esta clase se crean los métodos necesarios para que el usuario se registre, y se almacenan los datos en la Tabla Usuario de la base de datos creada.

**package** com.example.proyecto;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.content.ContentValues;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.EditText;  
  
**import** com.example.proyecto.entidades.AdminSQLiteOpenHelper;  
**import** com.example.proyecto.utilidades.Utilidades;  
**import** android.widget.Toast;  
  
**public class** RegistroUsuario **extends** AppCompatActivity {  
 **private** EditText **nombreUsuario**,**nombres**, **correo**, **ubicacion**, **telefono**, **contraseña**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_registro***);  
  
 **nombreUsuario**=(EditText)findViewById(R.id.***nombreUsuario***);  
 **nombres**=(EditText)findViewById(R.id.***nombre***);  
 **contraseña**=(EditText)findViewById(R.id.***contraseña***);  
 **correo**=(EditText)findViewById(R.id.***correo***);  
 **ubicacion**=(EditText)findViewById(R.id.***ubicacion***);  
 **telefono**= (EditText)findViewById(R.id.***telefono***);  
 }  
   
 *//ASIGNACIÓN DEL BOTÓN QUE REGISTRARÁ AL USUARIO* **public void** Registrar(View v) {  
 registrarUsuarios();  
 }  
  
*//MÉTODO QUE REGISTRA AL USUARIO* **private void** registrarUsuarios() {  
 AdminSQLiteOpenHelper admin= **new** AdminSQLiteOpenHelper(**this**,**"PST\_G6"**,**null**,1);  
 SQLiteDatabase db= admin.getWritableDatabase();  
  
 ContentValues values=**new** ContentValues();  
 values.put(Utilidades.***KEY\_USUARIO\_NOMBREUSUARIO***,**nombreUsuario**.getText().toString());  
 values.put(Utilidades.***KEY\_USUARIO\_NOMBRES***,**nombres**.getText().toString());  
 values.put(Utilidades.***KEY\_USUARIO\_CONTRASEÑA***,**contraseña**.getText().toString());  
 values.put(Utilidades.***KEY\_USUARIO\_CORREO***,**correo**.getText().toString());  
 values.put(Utilidades.***KEY\_USUARIO\_TELEFONO***,**telefono**.getText().toString());  
 values.put(Utilidades.***KEY\_USUARIO\_UBICACION***,**ubicacion**.getText().toString());  
  
  
 Long idResultante=db.insert(Utilidades.***TABLA\_USUARIO***,Utilidades.***KEY\_USUARIO\_ID***,values);  
  
 Toast.*makeText*(getApplicationContext(),**"Registro exitoso de "**+ **nombres**.getText().toString()+**"\n Id de Usuario:"**+idResultante,Toast.***LENGTH\_SHORT***).show();  
 db.close();  
 limpiar();  
 }  
  
 *// VUELVE AL LOGIN* **public void** Volver(View v) {  
 finish();  
 }  
   
 *//LIMPIA LOS CAMPOS REQUERIDOS EN EL REGISTRO* **private void** limpiar() {  
 **nombreUsuario**.setText(**""**);  
 **nombres**.setText(**""**);  
 **contraseña**.setText(**""**);  
 **correo**.setText(**""**);  
 **ubicacion**.setText(**""**);  
 **telefono**.setText(**""**);  
 }  
}

**Login.java**

Dentro de esta actividad se crean los métodos necesarios para validar que el usuario haya ingresado su nombre de usuario y contraseña correcta.

**package** com.example.proyecto;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.content.Intent;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** com.example.proyecto.entidades.AdminSQLiteOpenHelper;  
**import** com.example.proyecto.utilidades.Utilidades;  
  
**public class** Login **extends** AppCompatActivity {  
 **private** EditText **nombreUsuario**,**contraseña**;  
 AdminSQLiteOpenHelper **admin**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_login***);  
  
 **admin** = **new** AdminSQLiteOpenHelper(**this**,  
 **"PST\_G6"**, **null**, 1);  
  
 **nombreUsuario**=(EditText)findViewById(R.id.***nom***);  
 **contraseña**=(EditText)findViewById(R.id.***contr***);  
 }  
  
 **public void** Ingresar(View v) {  
 consultarUsuario();  
 }  
  
 *//Método que valida el login del usuario indicando que tanto el nombre del usuario como la contraseña deben ser correctas.* **private void** consultarUsuario() {  
 SQLiteDatabase db=**admin**.getReadableDatabase();  
 String[] parametros={**nombreUsuario**.getText().toString()};  
 String[] campos={Utilidades.***KEY\_USUARIO\_NOMBREUSUARIO***,Utilidades.***KEY\_USUARIO\_CONTRASEÑA***};  
 String[] parametros2={**contraseña**.getText().toString()};  
 **try** {  
 Cursor usuario =db.query(Utilidades.***TABLA\_USUARIO***,campos,Utilidades.***KEY\_USUARIO\_NOMBREUSUARIO***+**"=?"**,parametros,**null**,**null**,**null**);  
 usuario.moveToFirst();  
 **nombreUsuario**.setText(usuario.getString(0));  
  
 **try** {  
 Cursor contraseña =db.query(Utilidades.***TABLA\_USUARIO***,campos,Utilidades.***KEY\_USUARIO\_CONTRASEÑA***+**"=?"**,parametros2,**null**,**null**,**null**);  
 Intent i = **new** Intent(**this**, UsuarioAc.**class**);  
 i.putExtra(**"nombre"**, **nombreUsuario**.getText().toString());  
 contraseña.move(1);  
 **nombreUsuario**.setText(contraseña.getString(1));  
 usuario.close();  
 contraseña.close();  
  
 startActivity(i);  
 }  
 **catch** (Exception e){  
 Toast.*makeText*(getApplicationContext(),**"La contraseña esta incorrecta"**,Toast.***LENGTH\_LONG***).show();  
 }  
  
 }**catch** (Exception e){  
 Toast.*makeText*(getApplicationContext(),**"El usuario no existe"**,Toast.***LENGTH\_LONG***).show();  
 limpiar();  
 }  
  
 }  
  
 *//METODO QUE LIMPIA LOS CAMPOS DE ENTRADA DE TEXTO DEL LOGIN* **private void** limpiar() {  
 **nombreUsuario**.setText(**""**);  
 **contraseña**.setText(**""**);  
 }  
}

**UsuarioAc.java**

Dentro de esta actividad se detallan los métodos del usuario para interactuar con la aplicación, como eliminar su registro, registrar, una mascota.

**package** com.example.proyecto;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.content.Context;  
**import** android.content.Intent;  
  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.net.ConnectivityManager;  
**import** android.net.NetworkInfo;  
**import** android.os.AsyncTask;  
**import** android.os.Bundle;  
**import** android.os.Handler;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** com.example.proyecto.entidades.AdminSQLiteOpenHelper;  
**import** com.example.proyecto.utilidades.Utilidades;  
  
**public class** UsuarioAc **extends** AppCompatActivity {

*//Se crean las diferentes variables de estado de la actividad***private** TextView **bienvenida**;  
**private** String **nombreUsuario**;  
AdminSQLiteOpenHelper **admin**; *// Instancia de la base de datos.*  
@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_usuario\_ac***);  
 **admin** =**new** AdminSQLiteOpenHelper(**this**,**"PST\_G6"**,**null**,1);  
  
 Bundle bundle = getIntent().getExtras();

*//Se obtiene el nombre de usuario que se ingresó en el login para dar la bienvenida al usuario.*  
 **bienvenida**= (TextView) findViewById(R.id.***mensajeBienvenida***);  
 **bienvenida**.setText(**"Bienvenid@ "**+bundle.getString(**"nombre"**));  
 **nombreUsuario**=bundle.getString(**"nombre"**);  
  
}

*//Este método asigna las funciones de los botones de interacción del usuario registrado.*  
**public void** onClick(View view) {  
 Intent miIntent=**null**;  
 **switch** (view.getId()){  
 **case** R.id.***registroMascota***:  
 miIntent=**new** Intent(UsuarioAc.**this**,RegistroMascota.**class**);  
 **break**;  
 **case** R.id.***verMascota***:  
 miIntent=**new** Intent(UsuarioAc.**this**,ListaMascotas.**class**);  
 **break**;  
 **case** R.id.***eliminarUsuario***:  
 eliminarUsuario();  
 **break**;  
 **case** R.id.***horario***:  
 miIntent=**new** Intent(UsuarioAc.**this**,HorariodeComida.**class**);  
 **break**;  
 **case** R.id.***cerrarSesion***:  
 miIntent=**new** Intent(UsuarioAc.**this**,MainActivity.**class**);  
 **break**;  
 }  
 **if** (miIntent!=**null**){  
 startActivity(miIntent);  
 }  
}  
  
**private void** eliminarUsuario() {  
 SQLiteDatabase db=**admin**.getWritableDatabase();  
 String[] parametros={**nombreUsuario**};  
  
 db.delete(Utilidades.***TABLA\_USUARIO***,Utilidades.***KEY\_USUARIO\_NOMBREUSUARIO***+**"=?"**,parametros);  
 Toast.*makeText*(getApplicationContext(),**"Se Eliminó el usuario"**,Toast.***LENGTH\_LONG***).show();  
 db.close();  
 Intent i = **new** Intent(**this**, UsuarioAc.**class**);  
 startActivity(i);  
}

**ConsultarListaPersonas.java**

En esta actividad, se almacena en una lista, el registro de personas con su respectiva información, donde se muestra un listView con dicha lista.

**package** com.example.proyecto;  
  
**import** android.content.Intent;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.AdapterView;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.ListView;  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** com.example.proyecto.entidades.AdminSQLiteOpenHelper;  
**import** com.example.proyecto.entidades.Usuario;  
**import** com.example.proyecto.utilidades.Utilidades;  
  
**import** java.util.ArrayList;  
  
**public class** ConsultarListaPersonas **extends** AppCompatActivity {  
  
 ListView **listViewPersonas**;  
 ArrayList<String> **listaInformacion**;  
 ArrayList<Usuario> **listaUsuarios**;  
  
 AdminSQLiteOpenHelper **admin**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_consultar\_lista\_list\_view***);  
  
 **admin**=**new** AdminSQLiteOpenHelper(getApplicationContext(),**"PST\_G6"**,**null**,1);  
  
 **listViewPersonas**= (ListView) findViewById(R.id.***listViewPersonas***);  
  
 consultarListaPersonas();  
  
 ArrayAdapter adaptador=**new** ArrayAdapter(**this**,android.R.layout.***simple\_list\_item\_1***,**listaInformacion**);  
 **listViewPersonas**.setAdapter(adaptador);  
  
 **listViewPersonas**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> adapterView, View view, **int** pos, **long** l) {  
  
 Usuario user=**listaUsuarios**.get(pos);  
  
 Intent intent=**new** Intent(ConsultarListaPersonas.**this**,DetallesdelUsuario.**class**);  
  
 Bundle bundle=**new** Bundle();  
 bundle.putSerializable(**"usuario"**,user);  
  
 intent.putExtras(bundle);  
 startActivity(intent);  
 }  
 });  
  
 }  
 *//Lee la base de datos de la tabla USUARIO y los almacena en un arrayList el cual es de tipo Usuario* **public void** consultarListaPersonas() {  
 SQLiteDatabase db=**admin**.getReadableDatabase();  
  
 Usuario usuario=**null**;  
 **listaUsuarios**=**new** ArrayList<Usuario>();  
 Cursor cursor=db.rawQuery(**"SELECT \* FROM "**+ Utilidades.***TABLA\_USUARIO***,**null**);  
  
 **while** (cursor.moveToNext()){  
 usuario=**new** Usuario();  
 usuario.setId(cursor.getInt(0));  
 usuario.setNombreUsuario(cursor.getString(1));  
 usuario.setNombres(cursor.getString(2));  
 usuario.setContraseña(cursor.getString(3));  
 usuario.setCorreo(cursor.getString(4));  
 usuario.setUbicacion(cursor.getString(5));  
 usuario.setTelefono(cursor.getString(6));  
  
 **listaUsuarios**.add(usuario);  
 }  
 obtenerLista();  
 }  
 *//Obitiene la lista de las personas registradas* **public void** obtenerLista() {  
 **listaInformacion**=**new** ArrayList<String>();  
  
 **for** (**int** i=0; i<**listaUsuarios**.size();i++){  
 **listaInformacion**.add(**listaUsuarios**.get(i).getId()+**" - "** +**listaUsuarios**.get(i).getNombreUsuario());  
 }  
 }  
}

**DetallesdelUsuarios.java**

Esta actividad muestra toda la información del usuario previamente seleccionado en la actividad ConsultarListaPersonas.java

**package** com.example.proyecto;  
  
**import** android.os.Bundle;  
**import** android.widget.TextView;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** com.example.proyecto.entidades.Usuario;  
  
**public class** DetallesdelUsuario **extends** AppCompatActivity {  
 TextView **usuarioId**, **usuarioNombre**, **nombres**, **contraseña**,**email**,**ubicacion**, **telefono**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_detalle\_usuario***);  
  
 **usuarioId** = (TextView) findViewById(R.id.***idUser***);  
 **usuarioNombre** = (TextView) findViewById(R.id.***tvNombreUsuario***);  
 **nombres**= (TextView) findViewById(R.id.***tvNombre***);  
 **contraseña**= (TextView) findViewById(R.id.***tvContraseña***);  
 **email** = (TextView) findViewById(R.id.***tvEmail***);  
 **ubicacion** = (TextView) findViewById(R.id.***tvUbicacion***);  
 **telefono** = (TextView) findViewById(R.id.***tvTelefono***);  
  
 Bundle objetoEnviado=getIntent().getExtras();  
 Usuario user=**null**;  
  
 **if**(objetoEnviado!=**null**){  
 user= (Usuario) objetoEnviado.getSerializable(**"usuario"**);  
 **usuarioId**.setText(**"ID: "**+user.getId().toString());  
 **usuarioNombre**.setText(**"Nombre de usuario: "**+user.getNombreUsuario());  
 **nombres**.setText(**"Nombre : "**+user.getNombres());  
 **contraseña**.setText(**"Contraseña: "**+user.getContraseña());  
 **email**.setText(**"Correo: "**+user.getCorreo());  
 **telefono**.setText(**"Teléfono: "**+user.getTelefono());  
 **ubicacion**.setText(**"Ubicación: "**+user.getUbicacion());  
 }  
 }  
}

**Paso: Registrar Mascota y mostrar mascotas registradas**

**RegistrarMascota.java**

En esta actividad se registra la mascota con los campos de nombre, edad, raza y se selecciona un tamaño a través de un spinner.

**package** com.example.proyecto;  
  
**import** android.content.ContentValues;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.AdapterView;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.EditText;  
**import** android.widget.Spinner;  
**import** android.widget.Toast;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** com.example.proyecto.entidades.AdminSQLiteOpenHelper;  
**import** com.example.proyecto.entidades.Mascota;  
**import** com.example.proyecto.entidades.Usuario;  
**import** com.example.proyecto.utilidades.Utilidades;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**import static** com.example.proyecto.utilidades.Utilidades.***KEY\_MASCOTA\_ID***;  
**import static** com.example.proyecto.utilidades.Utilidades.***TABLA\_MASCOTA***;  
  
**public class** RegistroMascota **extends** AppCompatActivity {  
  
 EditText **razaMascota**,**nombreMascota**, **edadMascota**;  
 Spinner **spinnerTamaño**; *// este spinner sirve para mostrar el tamaño a seleccionar* List **tamañosList** = **new** ArrayList( );  
 String **tamañoEscogido**;  
  
 AdminSQLiteOpenHelper **admin**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_registro\_mascota***);  
  
 **razaMascota** = (EditText) findViewById(R.id.***raza***);  
 **nombreMascota** = (EditText) findViewById(R.id.***nombreMascota***);  
 **edadMascota** = (EditText) findViewById(R.id.***edadMascota***);  
 **spinnerTamaño**= (Spinner) findViewById(R.id.***spinnerTamaño***);*//para escoger el tamaño de la mascota* **admin**=**new** AdminSQLiteOpenHelper(getApplicationContext(),**"PST\_G6"**,**null**,1);  
  
 **tamañosList**.add(**"Seleccione un tamaño"**);  
 **tamañosList**.add(**"Miniatura"**);  
 **tamañosList**.add(**"Pequeño"**);  
 **tamañosList**.add(**"Mediano"**);  
 **tamañosList**.add(**"Grande"**);  
 **tamañosList**.add(**"Extra Grande"**);  
  
 *//Adapatador necesario para el spinner del tamaño* ArrayAdapter<CharSequence> adaptador=**new** ArrayAdapter  
 (**this**,android.R.layout.***simple\_spinner\_item***,**tamañosList**);  
  
 **spinnerTamaño**.setAdapter(adaptador);  
  
 **spinnerTamaño**.setOnItemSelectedListener(**new** AdapterView.OnItemSelectedListener() {  
 *//Este método es solo para seleccionar el tamaño y registrarlo* @Override  
 **public void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** idl) {  
 }  
  
 @Override  
 **public void** onNothingSelected(AdapterView<?> adapterView) {  
 }  
 });  
  
 }  
 *//se registra la mascota con los campos ingresados* **public void** registrarMascota() {  
  
 SQLiteDatabase db=**admin**.getWritableDatabase(); *// se llama a la base de datos para escribirla.  
 /\*\*  
 \* Valida la seleccion del combo de los tamaños, si el usuario elige "seleccione tamaño" entonces  
 \* se retorna el id 0 ya que la palabra "seleccione tamaño" se encuentra en la pos 0 del spinner,  
 \* caso contrario se retorna la posicion del spinner para consultar el usuario almacenado en la lista  
 \*/* **int** position= (**int**) **spinnerTamaño**.getSelectedItemId();  
  
 **if** (position!=0 ){  
 Log.*i*(**"TAMAÑO"**,**tamañosList**.size()+**""**);  
 Log.*i*(**"id combo"**,position+**""**);  
 Log.*i*(**"id combo - 1"**,(position-1)+**""**);*//se resta 1 ya que se quiere obtener la posicion de la lista, no del combo* Log.*i*(**"tamaño"**,**tamañoEscogido**+**""**);  
 **tamañoEscogido** =**tamañosList**.get(position-1).toString();  
 ContentValues values=**new** ContentValues();  
 values.put(Utilidades.***KEY\_MASCOTA\_NOMBRE***,**nombreMascota**.getText().toString());  
 values.put(Utilidades.***KEY\_RAZA***,**razaMascota**.getText().toString());  
 values.put(Utilidades.***KEY\_MASCOTA\_EDAD***,**edadMascota**.getText().toString());  
 values.put(Utilidades.***KEY\_TAMAÑO***,**tamañoEscogido**);  
  
 Long idResultante=db.insert(Utilidades.***TABLA\_MASCOTA***, ***KEY\_MASCOTA\_ID***,values);  
 Toast.*makeText*(getApplicationContext(),**"Su Mascota: "**+**nombreMascota**.getText().toString()+**" se registró con éxito"**+ idResultante,Toast.***LENGTH\_SHORT***).show();  
 db.close();  
 limpiar();  
 }  
 **else**{  
 Toast.*makeText*(getApplicationContext(),**"Debe seleccionar un Tamaño"**,Toast.***LENGTH\_LONG***).show();  
 }  
 }  
  
 **public void** onClick(View view) {  
 **switch** (view.getId()){  
 **case** R.id.***registroMascota2***:  
 registrarMascota();  
 }  
 }  
  
 **private void** limpiar() {  
 **razaMascota**.setText(**""**);  
 **nombreMascota**.setText(**""**);  
 **edadMascota**.setText(**""**);  
 }  
  
 **public** List<Mascota> ObtenerTodasLasMascotas() {  
 List<Mascota> mascotas = **new** ArrayList<>();  
  
 *// Se asocia la mascota con el usuario* String POSTS\_SELECT\_QUERY =  
 String.*format*(**"SELECT \* FROM TABLA\_MASCOTA LEFT OUTER JOIN TABLA\_USUARIO ON TABLA\_MASCOTA.KEY\_MASCOTA\_ID\_USUARIO = KEY\_USUARIO\_ID"**,  
 Utilidades.***TABLA\_MASCOTA***,  
 Utilidades.***TABLA\_USUARIO***,  
 Utilidades.***TABLA\_MASCOTA***,Utilidades.***KEY\_MASCOTA\_ID\_USUARIO*** ,  
 Utilidades.***TABLA\_USUARIO***, Utilidades.***KEY\_USUARIO\_ID***);  
  
 SQLiteDatabase db = **admin**.getReadableDatabase();  
 Cursor cursor = db.rawQuery(POSTS\_SELECT\_QUERY, **null**);  
 **try** {  
 **if** (cursor.moveToFirst()) {  
 **do** {  
 Usuario nuevoUsuario = **new** Usuario();  
 nuevoUsuario.setNombreUsuario(cursor.getString(cursor.getColumnIndex(Utilidades.***KEY\_USUARIO\_NOMBREUSUARIO***))) ;  
 Mascota nuevaMascota = **new** Mascota();  
 nuevaMascota.setNombreMascota(cursor.getString(cursor.getColumnIndex(Utilidades.***KEY\_MASCOTA\_NOMBRE***)));  
 nuevaMascota.**dueño**= nuevoUsuario;  
  
 mascotas.add(nuevaMascota);  
 } **while**(cursor.moveToNext());  
 }  
 } **catch** (Exception e) {  
 Toast.*makeText*(getApplicationContext(),**"Error al intentar obtener mascotras"**,Toast.***LENGTH\_LONG***).show();  
 } **finally** {  
 **if** (cursor != **null** && !cursor.isClosed()) {  
 cursor.close();  
 }  
 }  
 **return** mascotas;  
}  
  
}

**ListaMascotas.java**

En esta actividad se crea una lista con las mascotas registradas para mostrar en un ListView dicha lista.

**package** com.example.proyecto;  
  
**import** android.content.Intent;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.AdapterView;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.ListView;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** com.example.proyecto.entidades.AdminSQLiteOpenHelper;  
**import** com.example.proyecto.entidades.Mascota;  
**import** com.example.proyecto.entidades.Usuario;  
**import** com.example.proyecto.utilidades.Utilidades;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** ListaMascotas **extends** AppCompatActivity {  
 ArrayList<String> **listaInformacion**;  
 ArrayList<Mascota> **listaMascotas**; *//Se almacenan las mascotas de tipo Mascotas* ListView **listViewMascota**;  
 AdminSQLiteOpenHelper **admin**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_lista\_mascotas***);  
  
 *//Inicializacion de instancias* **admin**=**new** AdminSQLiteOpenHelper(getApplicationContext(),**"PST\_G6"**,**null**,1);  
 **listViewMascota**= (ListView) findViewById(R.id.***listViewMascotas***);  
  
 *//Se llama al método que consulta la lista d mascotas de la db* consultarListaPersonas();  
  
 *//Adapatador necesario para el Spinner de Mascota* ArrayAdapter adaptador=**new** ArrayAdapter(**this**,android.R.layout.***simple\_list\_item\_1***,**listaInformacion**);  
 **listViewMascota**.setAdapter(adaptador);  
  
 **listViewMascota**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> adapterView, View view, **int** pos, **long** l) {  
  
 Mascota mascota=**listaMascotas**.get(pos);  
  
 Intent intent=**new** Intent(ListaMascotas.**this**,DetalleMascotas.**class**);  
  
 Bundle bundle=**new** Bundle();  
 bundle.putSerializable(**"mascota"**,mascota); *//entrega la mascota para el otro activity: DetalleMascotas* intent.putExtras(bundle);  
 startActivity(intent);  
  
 }  
 });  
 }  
  
 **private void** consultarListaPersonas() {  
 SQLiteDatabase db=**admin**.getReadableDatabase();  
  
 **listaMascotas**=**new** ArrayList<Mascota>(); *// lista que almacenara las mascotas  
 //selecciona todos los parametros de una mascota en la base de datos* Cursor cursor=db.rawQuery(**"SELECT \* FROM "**+ Utilidades.***TABLA\_MASCOTA***,**null**);  
  
 **while** (cursor.moveToNext()){  
 Mascota mascota=**new** Mascota();  
 mascota.setIdMascota(cursor.getInt(0));  
 mascota.setNombreMascota(cursor.getString(3));  
 mascota.setRaza(cursor.getString(4));  
 mascota.setIdUsuario(cursor.getInt(2));  
  
 **listaMascotas**.add(mascota);  
 }  
 obtenerLista();  
 }  
  
 **private void** obtenerLista() {  
 **listaInformacion**=**new** ArrayList<String>();  
  
 **for** (**int** i=0; i<**listaMascotas**.size();i++){  
 **listaInformacion**.add(**listaMascotas**.get(i).getIdMascota()+**" - "** +**listaMascotas**.get(i).getNombreMascota());  
 }  
 }  
}

**DetalleMascota.java**

En esta actividad de guardan los detalles registrados de la lista de mascota, en donde su correspondiente layout muestra toda la información de una mascota al seleccionarla de la actividad anterior: ListaMascotas.java.

**package** com.example.proyecto;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** com.example.proyecto.entidades.AdminSQLiteOpenHelper;  
**import** com.example.proyecto.entidades.Mascota;  
**import** com.example.proyecto.utilidades.Utilidades;  
  
  
**public class** DetalleMascotas **extends** AppCompatActivity {  
  
 AdminSQLiteOpenHelper **admin**;  
 TextView **campoIdMascota**, **campoNombreMascota**, **campoRaza**,**campoIdPersona**, **campoNombrePersona**, **campotamaño**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_detalle\_mascota***);  
  
 **admin**=**new** AdminSQLiteOpenHelper(getApplicationContext(),**"PST\_G6"**,**null**,1);  
  
 **campoIdPersona** = (TextView) findViewById(R.id.***campoIdPersona***);  
 **campoNombrePersona** = (TextView) findViewById(R.id.***campoNombrePersona***);  
 **campoIdMascota** = (TextView) findViewById(R.id.***campoIdMascota***);  
 **campoNombreMascota** = (TextView) findViewById(R.id.***campoNombreMascota***);  
 **campoRaza** = (TextView) findViewById(R.id.***campoRaza***);  
 **campotamaño** = (TextView) findViewById(R.id.***tamaño***);  
  
 Bundle objetoEnviado=getIntent().getExtras();  
 Mascota mascota=**null**;  
  
 **if**(objetoEnviado!=**null**){  
 mascota= (Mascota) objetoEnviado.getSerializable(**"mascota"**);  
 **campoIdMascota**.setText(mascota.getIdMascota().toString());  
 **campoNombreMascota**.setText(mascota.getNombreMascota());  
 **campoRaza**.setText(mascota.getRaza());  
 **campotamaño**.setText((mascota.getTamaño()));  
 consultarPersona(mascota.getIdUsuario());  
 }  
 }

*//Este método consulta la persona con su mascota*

**private void** consultarPersona(Integer idPersona) {  
 SQLiteDatabase db=**admin**.getReadableDatabase();  
 String[] parametros={idPersona.toString()};  
 String[] campos={Utilidades.***KEY\_USUARIO\_NOMBREUSUARIO***,Utilidades.***KEY\_USUARIO\_CONTRASEÑA***};  
 Toast.*makeText*(getApplicationContext(),**"Id"**+idPersona,Toast.***LENGTH\_LONG***).show();  
 **try** {  
 Cursor cursor =db.query(Utilidades.***TABLA\_USUARIO***,campos,Utilidades.***KEY\_USUARIO\_ID***+**"=?"**,parametros,**null**,**null**,**null**);  
 cursor.moveToFirst();  
 **campoIdPersona**.setText(idPersona.toString());  
 **campoNombrePersona**.setText(cursor.getString(0));  
 cursor.close();  
 }**catch** (Exception e){  
 Toast.*makeText*(getApplicationContext(),**"El id del dueño no existe"**,Toast.***LENGTH\_LONG***).show();  
 **campoNombrePersona**.setText(**""**);  
  
 }  
 }  
}

**Paso: Actividad Inicial**

**MainActivity.java**

Esta actividad es la principal de nuestra aplicación, donde el usario puede registrarse y/o iniciar sesión, además se puede ver una lista de los usuarios registrados.

**package** com.example.proyecto;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** com.example.proyecto.entidades.AdminSQLiteOpenHelper;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 AdminSQLiteOpenHelper admin=**new** AdminSQLiteOpenHelper(**this**,**"PST\_G6"**,**null**,1); *//Se accede a la base de datos* }  
 *//metod que valida los botones del La actividad principal de la aplicación* **public void** onClick(View view) {  
 Intent miIntent=**null**;  
 **switch** (view.getId()){  
 **case** R.id.***registrarseInicio***:  
 miIntent=**new** Intent(MainActivity.**this**,RegistroUsuario.**class**);  
 **break**;  
 **case** R.id.***loginUsuario***:  
 miIntent=**new** Intent(MainActivity.**this**,Login.**class**);  
 **break**;  
 **case** R.id.***ConsultaListaUsuario***:  
 miIntent=**new** Intent(MainActivity.**this**, ConsultarListaPersonas.**class**);  
 **break**;  
  
 }  
 **if** (miIntent!=**null**){  
 startActivity(miIntent);  
 }  
  
}}

**Paso: Hora-Notificación**

**AndroidManifest.xml:**

Se otorgan los permisos necesarios a la aplicación para poder mostrar las notificaciones, realizar el evento de alarma y poder recibir eventos de reinicio. Se declaran los servicios y receptores que reaccionaran a los eventos.

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.proyecto"**>  
 <**uses-permission android:name="android.permission.INTERNET"** />  
 <**uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"**/>  
 <**uses-permission android:name="android.permission.WRITE\_CALENDAR"** />  
 <**uses-permission android:name="android.permission.READ\_CALENDAR"** />  
 <**uses-permission android:name="android.permission.FOREGROUND\_SERVICE"** />  
 <**uses-permission android:name="android.permission.WAKE\_LOCK"** />  
 <**uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"** />  
 <**uses-permission android:name="com.google.android.gms.permission.ACTIVITY\_RECOGNITION"** />  
 <**uses-permission android:name="android.permission.VIBRATE"** />  
 <**uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED"**/>  
 <**application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_icone"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_icone\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme"**>  
 <**activity android:name=".RegistroMascota"**/>  
 <**activity android:name=".RegistroUsuario"**/>  
 <**activity android:name=".UsuarioAc"**/>  
 <**activity android:name=".HorariodeComida"**/>  
 <**activity android:name=".DetalleMascotas"**/>  
 <**activity android:name=".ConsultarListaPersonas"**/>  
 <**activity android:name=".ListaMascotas"**/>  
 <**activity android:name=".Login"**/>  
 <**activity android:name=".DetallesdelUsuario"**/>  
 <**activity android:name=".MainActivity"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
 </**activity**>  
 <**receiver  
 android:name="com.example.proyecto.entidades.AlarmReceiver"  
 android:enabled="true"** />  
  
 <**service  
 android:name="com.example.proyecto.entidades.NotificationService"  
 android:enabled="true"** />  
  
 <**service android:name="com.example.proyecto.entidades.RebootServiceClass"**/>  
  
 <**receiver android:name="com.example.proyecto.entidades.MyRebootReceiver"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.BOOT\_COMPLETED"**/>  
 </**intent-filter**>  
 </**receiver**>  
 </**application**>  
  
</**manifest**>

**AlarmReceiver.java:**

Recibe y guarda una alarma y cuando salte el evento llama a la clase que se encarga de diseñar y mostrar la notificación.

**package** com.example.proyecto.entidades;  
  
**import** android.content.BroadcastReceiver;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.net.Uri;  
**import** android.util.Log;  
  
**import** androidx.core.content.ContextCompat;**public class** AlarmReceiver **extends** BroadcastReceiver {  
  
 @Override  
 **public void** onReceive(Context context, Intent intent) {  
 Intent service1 = **new** Intent(context, NotificationService.**class**);  
 service1.setData((Uri.*parse*(**"custom://"** + System.*currentTimeMillis*())));  
 ContextCompat.*startForegroundService*(context, service1 );  
 Log.*d*(**"WALKIRIA"**, **" ALARM RECEIVED!!!"**);

**MyRebootReceiver.java:**

Se activa si recibe un evento de reinicio y llama a un servicio que configura la alarma guardada previamente.

**package** com.example.proyecto.entidades;  
  
**import** android.content.BroadcastReceiver;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.os.Build;**public class** MyRebootReceiver **extends** BroadcastReceiver {  
  
 @Override  
 **public void** onReceive(Context context, Intent intent) {  
 Intent serviceIntent = **new** Intent(context, RebootServiceClass.**class**);  
 serviceIntent.putExtra(**"caller"**, **"RebootReceiver"**);  
 **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***O***) {  
 context.startForegroundService(serviceIntent);  
 } **else** {  
 context.startService(serviceIntent);  
 }  
 }  
}

**RebootServiceClass.java:**

Revisa la hora de la alarma para poder leer a qué hora debe sonar y activarla cuando se dispare el evento.

**package** com.example.proyecto.entidades;  
  
**import** android.app.IntentService;  
**import** android.app.Notification;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.SharedPreferences;  
  
**import** com.example.proyecto.R;**public class** RebootServiceClass **extends** IntentService {  
 **public** RebootServiceClass(String name) {  
 **super**(name);  
 startForeground(1, **new** Notification());  
 }  
  
 **public** RebootServiceClass() {  
 **super**(**"RebootServiceClass"**);  
 }  
  
 @Override  
 **public void** onCreate() {  
 **super**.onCreate();  
 }  
  
 @Override  
 *//Cuando obtiene el extra y es “Rebootreceiver”, va a leer el id de la alarma y su hora y reguardarla* **protected void** onHandleIntent(Intent intent) {  
 String intentType = intent.getExtras().getString(**"caller"**);  
 **if** (intentType == **null**) **return**;  
 **if** (intentType.equals(**"RebootReceiver"**)) {  
 SharedPreferences settings = getSharedPreferences(getString(R.string.***app\_name***), Context.***MODE\_PRIVATE***);  
 Utils.*setAlarm*(settings.getInt(**"alarmID"**, 0), settings.getLong(**"alarmTime"**, 0), **this**);  
 }  
 }  
}

**Utils.java:**

Recibe la información de la alarma y la guarda en AlarmaManager.

**package** com.example.proyecto.entidades;  
  
**import** android.app.AlarmManager;  
**import** android.app.PendingIntent;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.net.Uri;  
  
**import static** android.content.Context.***ALARM\_SERVICE***;  
*//Guarda la alarma en AlarmaManager***public class** Utils {  
  
 **public static void** setAlarm(**int** i, Long timestamp, Context ctx) {  
 AlarmManager alarmManager = (AlarmManager) ctx.getSystemService(***ALARM\_SERVICE***);  
 Intent alarmIntent = **new** Intent(ctx, AlarmReceiver.**class**);  
 PendingIntent pendingIntent;  
 pendingIntent = PendingIntent.*getBroadcast*(ctx, i, alarmIntent, PendingIntent.***FLAG\_ONE\_SHOT***);  
 alarmIntent.setData((Uri.*parse*(**"custom://"** + System.*currentTimeMillis*())));  
 alarmManager.set(AlarmManager.***RTC\_WAKEUP***, timestamp, pendingIntent);  
 }  
}

**NotificationService.java:**

Creará la notificación que se mostrara cuando se dispare el evento, además de darle un mensaje, el nombre de la app, un tono o vibración y un mini-logo.

**package** com.example.proyecto.entidades;  
  
**import** android.annotation.TargetApi;  
**import** android.app.IntentService;  
**import** android.app.Notification;  
**import** android.app.NotificationChannel;  
**import** android.app.NotificationManager;  
**import** android.app.PendingIntent;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.res.Resources;  
**import** android.graphics.BitmapFactory;  
**import** android.media.RingtoneManager;  
**import** android.net.Uri;  
**import** android.os.Build;  
  
**import** androidx.core.app.NotificationCompat;  
  
**import** com.example.proyecto.HorariodeComida;  
**import** com.example.proyecto.R;  
*//Muestra la notificacion en la barra de tareas, cuando según la hora establecida***public class** NotificationService **extends** IntentService {  
  
 **private** NotificationManager **notificationManager**;  
 **private** PendingIntent **pendingIntent**;  
 **private static int** *NOTIFICATION\_ID* = 1;  
 Notification **notification**;  
  
  
 **public** NotificationService(String name) {  
 **super**(name);  
 }  
  
 **public** NotificationService() {  
 **super**(**"SERVICE"**);  
 }  
  
 @TargetApi(Build.VERSION\_CODES.***O***)  
 @Override  
 **protected void** onHandleIntent(Intent intent2) {  
 String NOTIFICATION\_CHANNEL\_ID = getApplicationContext().getString(R.string.***app\_name***);  
 Context context = **this**.getApplicationContext();  
 **notificationManager** = (NotificationManager) context.getSystemService(Context.***NOTIFICATION\_SERVICE***);  
 Intent mIntent = **new** Intent(**this**, HorariodeComida.**class**);  
 Resources res = **this**.getResources();  
 Uri soundUri = RingtoneManager.*getDefaultUri*(RingtoneManager.***TYPE\_ALARM***);  
  
 String message = getString(R.string.***new\_notification***);  
  
 **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***O***) {  
 **final int** NOTIFY\_ID = 0; *// ID of notification* String id = NOTIFICATION\_CHANNEL\_ID; *// default\_channel\_id* String title = NOTIFICATION\_CHANNEL\_ID; *// Default Channel* PendingIntent pendingIntent;  
 NotificationCompat.Builder builder;  
 NotificationManager notifManager = (NotificationManager) context.getSystemService(Context.***NOTIFICATION\_SERVICE***);  
 **if** (notifManager == **null**) {  
 notifManager = (NotificationManager) context.getSystemService(Context.***NOTIFICATION\_SERVICE***);  
 }  
 **int** importance = NotificationManager.***IMPORTANCE\_HIGH***;  
 NotificationChannel mChannel = notifManager.getNotificationChannel(id);  
 *//Se encarga de dar un patron de vibración a la notificación* **if** (mChannel == **null**) {  
 mChannel = **new** NotificationChannel(id, title, importance);  
 mChannel.enableVibration(**true**);  
 mChannel.setVibrationPattern(**new long**[]{100, 200, 300, 400, 500, 400, 300, 200, 400});  
 notifManager.createNotificationChannel(mChannel);  
 }  
 *//Se crea el mensaje a mostrar en la notificacion y un icono* builder = **new** NotificationCompat.Builder(context, id);  
 mIntent.setFlags(Intent.***FLAG\_ACTIVITY\_CLEAR\_TOP*** | Intent.***FLAG\_ACTIVITY\_SINGLE\_TOP***);  
 pendingIntent = PendingIntent.*getActivity*(context, 0, mIntent, PendingIntent.***FLAG\_UPDATE\_CURRENT***);  
 builder.setContentTitle(getString(R.string.***app\_name***)).setCategory(Notification.***CATEGORY\_SERVICE***)  
 .setSmallIcon(R.drawable.***ic\_notification***) *// required* .setContentText(message)  
 .setLargeIcon(BitmapFactory.*decodeResource*(res, R.drawable.***ic\_notification***))  
 .setDefaults(Notification.***DEFAULT\_ALL***)  
 .setAutoCancel(**true**)  
 .setSound(soundUri)  
  
 .setContentIntent(pendingIntent)  
 .setVibrate(**new long**[]{100, 200, 300, 400, 500, 400, 300, 200, 400});  
 Notification notification = builder.build();  
 notifManager.notify(NOTIFY\_ID, notification);  
  
 startForeground(1, notification);  
  
 } **else** {  
 **pendingIntent** = PendingIntent.*getActivity*(context, 1, mIntent, PendingIntent.***FLAG\_UPDATE\_CURRENT***);  
 **notification** = **new** NotificationCompat.Builder(**this**)  
 .setContentIntent(**pendingIntent**)  
 .setSmallIcon(R.drawable.***ic\_notification***)  
 .setLargeIcon(BitmapFactory.*decodeResource*(res, R.drawable.***ic\_notification***))  
 .setSound(soundUri)  
 .setAutoCancel(**true**)  
 .setContentTitle(getString(R.string.***app\_name***)).setCategory(Notification.***CATEGORY\_SERVICE***)  
 .setContentText(message).build();  
 **notificationManager**.notify(*NOTIFICATION\_ID*, **notification**);  
 }  
 }  
}

**HorariodeComida.java:**

Se permite agregar y eliminar horas en los que se mandaran las notificaciones, además de guardarlas en la base de datos de horario y mostrarlas en tiempo real, como parámetro en caso de que se reinicie el sistema.

**package** com.example.proyecto;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.app.TimePickerDialog;  
**import** android.content.ContentValues;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.SharedPreferences;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.AdapterView;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.EditText;  
**import** android.widget.ListView;  
**import** android.widget.TextView;  
**import** android.widget.TimePicker;  
**import** android.widget.Toast;  
  
**import** com.example.proyecto.entidades.AdminSQLiteOpenHelper;  
**import** com.example.proyecto.entidades.Horario;  
**import** com.example.proyecto.entidades.Usuario;  
**import** com.example.proyecto.entidades.Utils;  
**import** com.example.proyecto.utilidades.Utilidades;  
  
**import** java.util.ArrayList;  
**import** java.util.Calendar;  
  
**import static** android.view.View.\*;  
*//Ventana que permite manejar a que hora se desea las notificaciones***public class** HorariodeComida **extends** AppCompatActivity {  
 **private** TextView **notificationsTime**;  
 **private int alarmID**=1;  
 **private** SharedPreferences **settings**;  
 **private** EditText **descripcion**;  
 ListView **listaH**;  
 ArrayList<String> **listaInformacion**;  
 ArrayList<Horario> **listahora**;  
 AdminSQLiteOpenHelper **admin**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 *//Se declaran las variables* setContentView(R.layout.***activity\_horariode\_comida***);  
 **settings** = getSharedPreferences(getString(R.string.***app\_name***), Context.***MODE\_PRIVATE***);  
  
 **listaH** = (ListView) findViewById(R.id.***ListaHora***);  
  
 **notificationsTime** = (TextView) findViewById(R.id.***notifications\_time***);  
  
 **admin** = **new** AdminSQLiteOpenHelper(**this**,**"PST\_G6"**,**null**,1);  
  
 **descripcion** = (EditText) findViewById(R.id.***edit***);  
 String hour, minute;  
 hour = **settings**.getString(**"hour"**, **""**);  
 minute = **settings**.getString(**"minute"**, **""**);  
 *//Se revisa si ya existen alarmas para poder mostrarlas en la misma ventana* consultarListahoras();  
 *//Crea un arreglo que mostrara informacion de cada alarma* ArrayAdapter adaptador=**new** ArrayAdapter(**this**,android.R.layout.***simple\_list\_item\_1***,**listaInformacion**);  
 **listaH**.setAdapter(adaptador);  
 **listaH**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> adapterView, View view, **int** pos, **long** l) {  
 String informacion= **listahora**.get(pos).getHora();  
 **notificationsTime**.setText(informacion);  
 }  
 });  
 }  
  
 **private void** consultarListahoras() {  
*//Se lee la base de datos y se consulta que horas hay existentes* SQLiteDatabase db=**admin**.getReadableDatabase();  
 Horario hora =**null**;  
 **listahora** =**new** ArrayList<Horario>();  
 Cursor cursor=db.rawQuery(**"SELECT \* FROM "**+ Utilidades.***TABLA\_HORARIO***,**null**);  
  
 **while** (cursor.moveToNext()){  
 hora=**new** Horario();  
 hora.setId(cursor.getInt(0));  
 hora.setHora(cursor.getString(1));  
 hora.setDescripcion(cursor.getString(2));  
  
  
 **listahora**.add(hora);  
 }  
 obtenerLista();  
 }  
 **public void** obtenerLista() {  
 *//Se crea una lista con informacion sobre la alarma* **listaInformacion** = **new** ArrayList<String>();  
  
 **for** (**int** i = 0; i < **listahora**.size(); i++) {  
 **listaInformacion**.add(**listahora**.get(i).getHora()+**" - "**+**listahora**.get(i).getDescripcion());  
 }  
 }  
 **public void** eliminarUsuario(View view) {  
 *//Se elimina la alarma que el usuario haya escogido , y se vuelve a cargar el activity para mostrar el cambio* SQLiteDatabase db=**admin**.getWritableDatabase();  
 String[] parametros={**notificationsTime**.getText().toString()};  
  
 db.delete(Utilidades.***TABLA\_HORARIO***,Utilidades.***KEY\_HORARIO\_HORA***+**"=?"**,parametros);  
 Toast.*makeText*(getApplicationContext(),**"Se eliminó el recordatorio "**,Toast.***LENGTH\_LONG***).show();  
 db.close();  
 cargar();  
 }  
 **public void** agregarUsuario(View view) {  
 *//Utiliza la clase calendar para poder mostrar y obterner hora y minutos que se desea la alarma* Calendar mcurrentTime = Calendar.*getInstance*();  
 **int** hour = mcurrentTime.get(Calendar.***HOUR\_OF\_DAY***);  
 **int** minute = mcurrentTime.get(Calendar.***MINUTE***);  
 TimePickerDialog mTimePicker;  
 mTimePicker = **new** TimePickerDialog(HorariodeComida.**this**, **new** TimePickerDialog.OnTimeSetListener() {  
 @Override  
 **public void** onTimeSet(TimePicker timePicker, **int** selectedHour, **int** selectedMinute) {  
 String finalHour, finalMinute=**"0"**;  
 *//Se obtienen las variables del tiempo y se procede a mostrarla* finalHour = **""** + selectedHour;  
 finalMinute = **""** + selectedMinute;  
 **if** (selectedHour < 10) finalHour = **"0"** + selectedHour;  
 **if** (selectedMinute < 10) finalMinute = **"0"** + selectedMinute;  
 **notificationsTime**.setText(finalHour + **":"** + finalMinute);  
  
 Calendar today = Calendar.*getInstance*();  
  
 today.set(Calendar.***HOUR\_OF\_DAY***, selectedHour);  
 today.set(Calendar.***MINUTE***, selectedMinute);  
 today.set(Calendar.***SECOND***, 0);  
  
 SharedPreferences.Editor edit = **settings**.edit();  
 edit.putString(**"hour"**, finalHour);  
 edit.putString(**"minute"**, finalMinute);  
  
 *//Se guarda la hora seleccionada para que se cree la alarma si se reinicia el sistema* edit.putInt(**"alarmID"**, **alarmID**);  
 edit.putLong(**"alarmTime"**, today.getTimeInMillis());  
  
 edit.commit();  
 *//Se guarda la alarma en la base de datos* SQLiteDatabase db= **admin**.getWritableDatabase();  
  
 ContentValues values=**new** ContentValues();  
 values.put(Utilidades.***KEY\_HORARIO\_HORA***,finalHour + **":"** + finalMinute);  
 values.put(Utilidades.***KEY\_HORARIO\_DESCRIPCION***,**descripcion**.getText().toString());  
  
  
 Long idResultante=db.insert(Utilidades.***TABLA\_HORARIO***,Utilidades.***KEY\_HORARIO\_ID***,values);  
  
 Toast.*makeText*(getApplicationContext(),**"Alarma registrada "**+idResultante+**"\nHora: "**+finalHour + **":"** + finalMinute,Toast.***LENGTH\_SHORT***).show();  
 db.close();  
 *//Se programa la notificacion acorde la alarma* Utils.*setAlarm*(**alarmID**, today.getTimeInMillis(), HorariodeComida.**this**);  
 **if** (finalHour != **"0"** && finalMinute != **"0"**){  
 cargar();  
 }  
  
 }  
 }, hour, minute, **true**);*//Yes 24 hour time* mTimePicker.setTitle(getString(R.string.***select\_time***));  
 mTimePicker.show();  
  
 }  
 **private void** cargar() {  
 *//Vuelve a cargar el activity para que se muestre los cambios* overridePendingTransition(0, 0);  
 finish();  
 overridePendingTransition(0, 0);  
 startActivity(getIntent());  
 overridePendingTransition(0, 0);  
 }  
}

**Paso conexión Android-Arduino**

**Android Studio**

**Clase Conexion**

Esta clase es la encargada de hacer la conexión respectiva con el localhost definido, pero a su vez recolectar toda la información HTML que esta página web contenga.

import java.io.BufferedInputStream;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.net.HttpURLConnection;  
import java.net.URL;  
  
import javax.net.ssl.HttpsURLConnection;  
  
public class Conexion {  
 //se crea una variable tipo String de datos para guardar la informacion que existe en el localHost  
 private String datos= null;  
 //se crea un metodo que permite hacer la conexion con el localHost que a su vez esta conectado a arduino  
 public String GetArduino(String urlString){  
  
 try{  
 URL url= new URL(urlString);  
 //Se conecta a la URL designada en el constructor del método  
 HttpURLConnection httpsURLConnection = (HttpURLConnection) url.openConnection();  
 if(httpsURLConnection.getResponseCode() == 200){  
 //Se empieza a recolectar toda la informacion que existe en el LocalHost, codigo HTML  
 InputStream inputStream= new BufferedInputStream(httpsURLConnection.getInputStream());  
  
 BufferedReader bufferedReader= new BufferedReader(new InputStreamReader(inputStream,"UTF-8"));  
  
 StringBuilder stringBuilder= new StringBuilder();  
  
 String linea;  
 //Se recorre la informacion linea a linea y se lo almacena en un variable de titpo StringBuilder  
 while((linea= bufferedReader.readLine()) != null){  
 stringBuilder.append(linea);  
 }  
 //Se copia toda la informacion guardad en la variable stringBuilder en la variable datos  
 datos= stringBuilder.toString();  
 //se desconecta la conexion con el LocalHost  
 httpsURLConnection.disconnect();  
 }  
  
 }catch (IOException io){  
 return null;  
 }  
 //este metodo retorna la informacion que se encuentra en la variable datos  
 return datos;  
 }  
}

**Clase UsuarioAc**

Esta parte de la clase UsuarioAc, es la que permite la conectividad directa entre los botones que se encuentran en el localhost mandando señales al arduino, con los botones que se encuentran en la aplicación.

import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Context;  
import android.content.Intent;  
  
import android.database.sqlite.SQLiteDatabase;  
import android.net.ConnectivityManager;  
import android.net.NetworkInfo;  
import android.os.AsyncTask;  
import android.os.Bundle;  
import android.os.Handler;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.proyecto.entidades.AdminSQLiteOpenHelper;  
import com.example.proyecto.utilidades.Utilidades;  
  
public class UsuarioAc extends AppCompatActivity {  
  
 //Se crean las diferentes variables de estado de la actividad  
 private TextView bienvenida;  
 private String nombreUsuario;  
 AdminSQLiteOpenHelper admin;  
 Button btn1;  
 Button btn2;  
 TextView txtresultado;  
 boolean servo1= false;  
 boolean servo2= false;  
 final Handler handler = new Handler();  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_usuario\_ac);  
 admin =new AdminSQLiteOpenHelper(this,"PST\_G6",null,1);  
  
 //Se instancian los botonees creados en el XML  
 btn1= (Button)findViewById(R.id.comida);  
 btn2= (Button)findViewById(R.id.agua);  
 //txtresultado=(TextView)findViewById(R.id.txtResultado);  
 //solicitud();  
 //Comando para actualizar la aplicacion y el localhost  
 handler.postDelayed(actualizacion,0);  
  
 //se crea la acciones de los botones  
 //asiganciones de valor a el contrsuctor del metodo solicitud  
 btn1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 solicitud("?H2");  
 servo1= true;  
 }  
 });  
 btn2.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 solicitud("?H3");  
 servo2= true;  
 }  
 });  
  
 }  
  
  
 //Metodo creado para la conectividad con las diferentes extensiones web que posee el localhost  
 public void solicitud(String comando){  
 //Se conecta al servidor  
 ConnectivityManager connMgr= (ConnectivityManager)getSystemService(Context.CONNECTIVITY\_SERVICE);  
 NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();  
  
 //Nombre de la ip mas la extension del localhost  
 String url= "http://192.168.0.20/"+comando;  
 if(networkInfo != null && networkInfo.isConnected()) {  
 //Conexion directa a la extension  
 new DownloadWebpageTask().execute(url);  
 }else{  
 //txtresultado.setText("Conexion detectada");  
 }  
 }  
  
 //Hilo que actualiza la aplicacion cada 2 segundo  
 private Runnable actualizacion= new Runnable() {  
 @Override  
 public void run() {  
 solicitud("");  
 handler.postDelayed(this, 2000);  
 }  
 };  
  
 //metodo que permite crear la conexion directa desde la clase Conexion  
 private class DownloadWebpageTask extends AsyncTask<String,Void,String> {  
 @Override  
 protected String doInBackground(String... urls) {  
 Conexion conexion= new Conexion();  
  
 return conexion.GetArduino(urls[0]);  
 }  
 //@Override  
 protected void onPostExecute(String result) {  
 if(result!= null){  
 if(result.contains("Switch ON - Pin2") && servo1==true){  
 //btn1.setText("ACTIVADO");  
 System.out.println("hola1");  
 servo1= false;  
 //handler.postDelayed(actualizacion,0);  
 }  
 if (result.contains("Switch ON - Pin3") && servo2==true){  
 //btn2.setText("ACTIVADO");  
 System.out.println("hola2");  
 servo2= false;  
 //handler.postDelayed(actualizacion,0);  
 }else {  
 //txtresultado.setText("Conexion a la red");  
 }  
 } //txtresultado.setText(result);  
 }  
 }  
}

**AndroidManifest**

En AndroidManifest solo se crean dos permisos que son para internet

<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

**Arduino**

Este es el código que se encarga de la conexión entre arduino y el localhost, en el cual por medio de código de html, se crean los botones y se personaliza un poco la página.

//incluimos librerias para conexion ethernet

#include <**Ethernet**.h>

#include <**SPI**.h>

// Incluímos la librería para poder controlar el servo

#include <**Servo**.h>

// Declaramos la variable para controlar el servo

**Servo** servoMotor1;

**Servo** servoMotor2;

////////////////////////////////////////////////////////////////////////

//CONFIGUACION PST PROYECTO

////////////////////////////////////////////////////////////////////////

 //CONFIGURACION DE IP MANUAL

 byte ip[] = { 192, 168, 0, 20 };   //Manual setup only

 byte gateway[] = { 192, 168, 0, 1 }; //Manual setup only

 byte subnet[] = { 255, 255, 255, 0 }; //Manual setup only

 // MAC DE LA TARJETA SHIELD ETHERNET

 byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

 //Ethernet Port

**EthernetServer** server = **EthernetServer**(80); //default html port 80

 String readString;

 //NUMERO DE SERVOMOTORES A UTILIZAR

 int outputQuantity = 2;

 //El pin de salida más bajo desde el que estamos comenzando

////////////////////////////////////////////////////////////////////////

 // declaracion de variables

 int outputLowes = 0;

 boolean printLastCommandOnce = false;

 boolean printButtonMenuOnce = false;

 boolean initialPrint = true;

 String allOn = "";

 String allOff = "";

 boolean reading = false;

 boolean readInput[10]; //Cree una matriz booleana para la cantidad máxima.

//Inicio del programa

void setup(){

 pinMode(5, OUTPUT); //pin selector para control

 servoMotor1.attach(7);  //pin 6 control servo 1

 servoMotor2.attach(6);  //pin 7 control servo 2

 //iniciar ethernet

 // Iniciamos el monitor serial

**Serial**.begin(9600);

 //Pins 10,11,12 & 13 son usados para la shield ethernet

 //pines de salida

 for (int var = outputLowest; var < outputLowest + outputQuantity; var++)  {

           pinMode(var, OUTPUT);

       }

 //configuracion de direcciones ip

 //Ethernet.begin(mac); //for DHCP address. (Address will be printed to serial.)

**Ethernet**.begin(mac, ip, gateway, subnet); //configuracion manual. (Address is the one configured above.)

 server.begin();

**Serial**.println(**Ethernet**.localIP());

}

void loop(){

 // toda la parte del cliente y el mando de ejecucion para los servomotores

 checkForClient();

}

void checkForClient(){

**EthernetClient** client = server.available();

 if (client) {

   // una solicitud http termina con una línea en blanco

   boolean currentLineIsBlank = true;

   boolean sentHeader = false;

   while (client.connected()) {

     if (client.available()) {

       if(!sentHeader){

        // enviar un encabezado de respuesta http estándar

         client.println("HTTP/1.1 200 OK");

         client.println("Content-Type: text/html");

         client.println("Connnection: close");

         client.println();

         client.println("<!DOCTYPE HTML>");

         client.println("<head>");

         // agregar la pagina del titulo

         client.println("<title>PETFESHER</title>");

         client.println("<meta name=\"description\" content=\"PETFESHER\"/>");

         // añadir una etiqueta de actualización meta, por lo que el navegador se refresca cada 2 segundos:

         client.println("<html>");

         client.println("<meta http-equiv=\"refresh\" content=\"2.URL=http://192.168.0.20\">");

         client.println("</html>");

         // agregar otra configuración de navegador

         client.println("<meta name=\"apple-mobile-web-app-capable\" content=\"yes\">");

         client.println("<meta name=\"apple-mobile-web-app-status-bar-style\" content=\"default\">");

         client.println("<meta name=\"viewport\" content=\"width=device-width, user-scalable=no\"/>");

         //insertando los datos de estilos, que generalmente se encuentran en archivos CSS.

         client.println("<style type=\"text/css\">");

         client.println("");

         //Esto establecerá cómo se verá la página gráficamente

         client.println("html { height:100%; }");

         client.println("  body {");

         client.println("    height: 100%;");

         client.println("    margin: 0;");

         client.println("    font-family: helvetica, sans-serif;");

         client.println("    -webkit-text-size-adjust: none;");

         client.println("   }");

         client.println("");

         client.println("body {");

         client.println("    -webkit-background-size: 100% 21px;");

         client.println("    background-color: #c5ccd3;");

         client.println("    background-image:");

         client.println("    -webkit-gradient(linear, left top, right top,");

         client.println("    color-stop(.75, transparent),");

         client.println("    color-stop(.75, rgba(255,255,255,.1)) );");

         client.println("    -webkit-background-size: 7px;");

         client.println("   }");

         client.println("");

         client.println(".view {");

         client.println("    min-height: 100%;");

         client.println("    overflow: auto;");

         client.println("   }");

         client.println("");

         client.println(".header-wrapper {");

         client.println("    height: 44px;");

         client.println("    font-weight: bold;");

         client.println("    text-shadow: rgba(0,0,0,0.7) 0 -1px 0;");

         client.println("    border-top: solid 1px rgba(255,255,255,0.6);");

         client.println("    border-bottom: solid 1px rgba(0,0,0,0.6);");

         client.println("    color: #fff;");

         client.println("    background-color: #8195af;");

         client.println("    background-image:");

         client.println("    -webkit-gradient(linear, left top, left bottom,");

         client.println("    from(rgba(255,255,255,.4)),");

         client.println("    to(rgba(255,255,255,.05)) ),");

         client.println("    -webkit-gradient(linear, left top, left bottom,");

         client.println("    from(transparent),");

         client.println("    to(rgba(0,0,64,.1)) );");

         client.println("    background-repeat: no-repeat;");

         client.println("    background-position: top left, bottom left;");

         client.println("    -webkit-background-size: 100% 21px, 100% 22px;");

         client.println("    -webkit-box-sizing: border-box;");

         client.println("   }");

         client.println("");

         client.println(".header-wrapper h1 {");

         client.println("    text-align: center;");

         client.println("    font-size: 20px;");

         client.println("    line-height: 44px;");

         client.println("    margin: 0;");

         client.println("   }");

         client.println("");

         client.println(".group-wrapper {");

         client.println("    margin: 9px;");

         client.println("    }");

         client.println("");

         client.println(".group-wrapper h2 {");

         client.println("    color: #4c566c;");

         client.println("    font-size: 17px;");

         client.println("    line-height: 0.8;");

         client.println("    font-weight: bold;");

         client.println("    text-shadow: #fff 0 1px 0;");

         client.println("    margin: 20px 10px 12px;");

         client.println("   }");

         client.println("");

         client.println(".group-wrapper h3 {");

         client.println("    color: #4c566c;");

         client.println("    font-size: 12px;");

         client.println("    line-height: 1;");

         client.println("    font-weight: bold;");

         client.println("    text-shadow: #fff 0 1px 0;");

         client.println("    margin: 20px 10px 12px;");

         client.println("   }");

         client.println("");

         client.println(".group-wrapper table {");

         client.println("    background-color: #fff;");

         client.println("    -webkit-border-radius: 10px;");

         client.println("    -moz-border-radius: 10px;");

         client.println("    -khtml-border-radius: 10px;");

         client.println("    border-radius: 10px;");

         client.println("    font-size: 17px;");

         client.println("    line-height: 20px;");

         client.println("    margin: 9px 0 20px;");

         client.println("    border: solid 1px #a9abae;");

         client.println("    padding: 11px 3px 12px 3px;");

         client.println("    margin-left:auto;");

         client.println("    margin-right:auto;");

         client.println("    -moz-transform :scale(1);"); //Code for Mozilla Firefox

         client.println("    -moz-transform-origin: 0 0;");

         client.println("   }");

         client.println("");

         //cómo se verá el LED verde (encendido)

         client.println(".green-circle {");

         client.println("    display: block;");

         client.println("    height: 23px;");

         client.println("    width: 23px;");

         client.println("    background-color: #0f0;");

         client.println("    background-color: rgba(60, 132, 198, 0.8);");

         client.println("    -moz-border-radius: 11px;");

         client.println("    -webkit-border-radius: 11px;");

         client.println("    -khtml-border-radius: 11px;");

         client.println("    border-radius: 11px;");

         client.println("    margin-left: 1px;");

         client.println("    background-image: -webkit-gradient(linear, 0% 0%, 0% 90%, from(rgba(46, 184, 0, 0.8)), to(rgba(148, 255, 112, .9)));@");

         client.println("    border: 2px solid #ccc;");

         client.println("    -webkit-box-shadow: rgba(11, 140, 27, 0.5) 0px 10px 16px;");

         client.println("    -moz-box-shadow: rgba(11, 140, 27, 0.5) 0px 10px 16px; /\* FF 3.5+ \*/");

         client.println("    box-shadow: rgba(11, 140, 27, 0.5) 0px 10px 16px; /\* FF 3.5+ \*/");

         client.println("    }");

         client.println("");

         //cómo se verá el LED negro (apagado)

         client.println(".black-circle {");

         client.println("    display: block;");

         client.println("    height: 23px;");

         client.println("    width: 23px;");

         client.println("    background-color: #040;");

         client.println("    -moz-border-radius: 11px;");

         client.println("    -webkit-border-radius: 11px;");

         client.println("    -khtml-border-radius: 11px;");

         client.println("    border-radius: 11px;");

         client.println("    margin-left: 1px;");

         client.println("    -webkit-box-shadow: rgba(11, 140, 27, 0.5) 0px 10px 16px;");

         client.println("    -moz-box-shadow: rgba(11, 140, 27, 0.5) 0px 10px 16px; /\* FF 3.5+ \*/");

         client.println("    box-shadow: rgba(11, 140, 27, 0.5) 0px 10px 16px; /\* FF 3.5+ \*/");

         client.println("    }");

         client.println("");

         //esto agregará el deslumbramiento a ambos LED

         client.println("   .glare {");

         client.println("      position: relative;");

         client.println("      top: 1;");

         client.println("      left: 5px;");

         client.println("      -webkit-border-radius: 10px;");

         client.println("      -moz-border-radius: 10px;");

         client.println("      -khtml-border-radius: 10px;");

         client.println("      border-radius: 10px;");

         client.println("      height: 1px;");

         client.println("      width: 13px;");

         client.println("      padding: 5px 0;");

         client.println("      background-color: rgba(200, 200, 200, 0.25);");

         client.println("      background-image: -webkit-gradient(linear, 0% 0%, 0% 95%, from(rgba(255, 255, 255, 0.7)), to(rgba(255, 255, 255, 0)));");

         client.println("    }");

         client.println("");

         //y finalmente este es el final de los datos de estilo y el encabezado

         client.println("</style>");

         client.println("</head>");

         //now printing the page itself

         client.println("<body>");

         client.println("<div class=\"view\">");

         client.println("    <div class=\"header-wrapper\">");

         client.println("      <h1>Ethernet Switching</h1>");

         client.println("    </div>");

         client.println("<div  class=\"group-wrapper\">");

         client.println("    <h2>Switch the required output.</h2>");

         client.println();

         //Esto es para que el arduino construya la página sobre la marcha.

         sentHeader = true;

       }

       char c = client.read();

       //leer char por char solicitud HTTP

       if (readString.length() < 100) {

         //store characters to string

         readString += c;

         //Serial.print(c);

       }

       if(c == '?') {

         reading = true; //encontró el?, comience a leer la información

       }

       if(reading && c == ' '){

         reading = false;

       }

//       Serial.print(c);

       //condiciones de manejo de servomotores

       if(reading){

         //servomotor1

         if(c== '2')

           {

             //comunicacion del servo comida

             servoMotor1.write(10);

             delay(3000);

             servoMotor1.write(110);

           }

        //servomotor2

        if(c=='3')

           {

             //comunicacion del servo agua

             servoMotor2.write(10);

             delay(3000);

             servoMotor2.write(110);

           }

       //borrando cadena para la próxima lectura

         readString="";

         }

       //print de los botones

       if (c == '\n' && currentLineIsBlank){

**Serial**.println("HOLA");

         printLastCommandOnce = true;

         printButtonMenuOnce = true;

         if (printButtonMenuOnce == true){

              //Start to create the html table

    client.println("");

    //client.println("<p>");

    client.println("<FORM>");

    client.println("<table border=\"0\" align=\"center\">");

              printHtmlButtons(client);

              printButtonMenuOnce = false;

         }

         break;

       }

     }

   }

   //Establecer variables antes de salir

   printLastCommandOnce = false;

   printButtonMenuOnce = false;

   allOn = "";

   allOff = "";

   client.println("\n<h3 align=\"center\">&copy; ProyectoPST <br> GRUPO 6-ESPOL</h3>");

   client.println("</div>\n</div>\n</body>\n</html>");

   delay(1); // dar tiempo al navegador web para recibir los datos

   client.stop(); // cerrar la connection:

 }

}

//impresion de los botones en html

void printHtmlButtons(**EthernetClient** client){

    //Comience a crear la tabla html

    client.println("");

    //client.println("<p>");

    client.println("<FORM>");

    client.println("<table border=\"0\" align=\"center\">");

    //Iniciar la impresión botón por botón

    for (int var = outputLowest; var < outputLowest + outputQuantity; var++)  {

             allOn += "H";

             allOn += var;

             //Imprimir inicio de fila

             client.print("<tr>\n");

             //Imprime los botones ON

             client.print(" <td><INPUT TYPE=\"button\" VALUE=\"Switch ON - Pin");

             client.print(var);

             client.print("\" onClick=\"parent.location='/?H");

             client.print(var);

             client.print("'\"></td>\n");

             //Imprimir el final de la fila

             client.print("</tr>\n");

        }

             //Cerrando el table y form

             client.println("</table>");

             client.println("</FORM>");

             //client.println("</p>");

   }

//Lee las salidas de los estados

void readOutputStatuses(){

 for (int var = outputLowest; var < outputLowest + outputQuantity; var++)  {

           readInput[var] = digitalRead(var);

      }

}

1. **Análisis de presupuesto**

El presupuesto de nuestro proyecto fue el siguiente:

|  |  |  |
| --- | --- | --- |
| Nombre | Cantidad | Precio |
| Servomotor | 2 | $11 |
| Módulo Ethernet | 1 | $11.5 |
| Otros | - | $5 |
| TOTAL |  | $38.5 |

Por lo cual el presupuesto de nuestro proyecto fue de 38.5, a este no se le suma el costo del arduino debido a que ya se disponía de este, pero si se lo adjuntara el valor subiría a los $ 58.5, por lo cual sería un proyecto de costo intermedio.

1. **Conclusiones**

* La aplicación logra ayudar a los usuarios a mantener un recordatorio de los momentos de alimentar a sus mascotas, y poder hacerlo a distancia.
* La aplicación móvil desarrollada programa una notificación de alimentar a la mascota y permite que el dispensador proporcione la comida/agua por medio de botones por lo cual no es necesario que el usuario deba proporcional la comida por sí mismo, mientras los depósitos no estén vacíos, se manejara a distancia.
* Los entornos integrados de desarrollo (IDE) facilitaron la elaboración de esta aplicación, debido a que ofrece herramientas de edición, compilación, depuración, análisis y ejecución en una misma interfaz para que los archivos xml creen formatos de interfaces amigables e intuitivos para el usuario.
* La utilización de notificaciones permite que el usuario reciba la información sin necesidad de tener activa la aplicación. Es decir que la aplicación está en un segundo plano y solo realiza el evento a la hora que se estableció.

1. **REFERENCIAS BIBLIOGRÁFICAS**

* Osorio-Carmona, E., Giraldo-Carmona, J., & Narváez-Solarte, W. (2012). Metodologías para determinar la digestibilidad de los alimentos utilizados en la alimentación canina. *vet. zootec*, *6*(1), 87-97.
* Valdés, F., & Areny, R. P. (2007). *Microcontroladores fundamentos y aplicaciones con PIC* (Vol. 1149). Marcombo.
* Sl Didact. (2005). *Manual de programacion lenguaje C++*. MAD-Eduforma.
* Groussard, T. (2012). *JAVA 7: Los fundamentos del lenguaje Java*. Ediciones Eni.
* Dauvergne, C. (2018). *La alimentación del perro*. Parkstone International.
* Rountev, A., & Yan, D. (2014, February). Static reference analysis for GUI objects in Android software. In *Proceedings of Annual IEEE/ACM International Symposium on Code Generation and Optimization* (pp. 143-153).
* Arduino, S. A. (2015). Arduino. *Arduino LLC*.
* Yedamale, P. (2003). Brushless DC (BLDC) motor fundamentals. *Microchip Technology Inc*, *20*, 3-15.