UNIVERSIDAD TECNOLÓGICA DEL PERÚ



FILIAL AREQUIPA

CIENCIAS E INGENIERÍAS

INGENIERÍA DE SISTEMAS E INFORMÁTICA

“SOFTWARE PARA CONTROLAR EL ABASTECIMIENTO DE COMBUSTIBLE”

Integrante:

Percy Saico ccapa

Profesor Asesor:

Ing. Yovani Quinteros Camapaza

AREQUIPA – PERÚ

2014

# Contenido

Contenido

[Contenido 3](#_Toc397349415)

[Introducción 5](#_Toc397349416)

[Lista de figuras 6](#_Toc397349417)

[1.1 Descripción del Problema 7](#_Toc397349418)

[1.3 Obtención de los Requerimientos 8](#_Toc397349419)

[1.3.1 Requerimientos Funcionales. 8](#_Toc397349420)

[1.3.2 Requerimientos no Funcionales. 9](#_Toc397349421)

[1.4 Arquitectura del Software 10](#_Toc397349422)

[1.4.1 Diagramas de Clases 10](#_Toc397349423)

[1.4.1.1 Diagrama de Clases de la Capa de Datos 11](#_Toc397349424)

[1.4.1.2 Diagrama de Clases de la Capa Entidad 12](#_Toc397349425)

[1.4.1.3 Diagrama de Clases de la Capa de Negocio 13](#_Toc397349426)

[1.4.1.4 Diagrama de Clases de la Capa UI 14](#_Toc397349427)

[1.4.2 Diagramas de la Base de Datos 18](#_Toc397349428)

[1.4.2.1 Diagrama Lógico de la Base de datos 18](#_Toc397349429)

[1.4.2.2 Tablas 19](#_Toc397349430)

[1.4.2.3 Secuencias 22](#_Toc397349431)

[1.4.2.4 Índices 22](#_Toc397349432)

[1.4.2.5 Vistas 24](#_Toc397349433)

[1.4.2.6 Procedimientos Almacenados 25](#_Toc397349434)

[1.4.2.7 Desencadenadores 28](#_Toc397349435)

[1.4.3 Patrón de Diseño 31](#_Toc397349436)

[1.4.3.1 Programación en N Capas 31](#_Toc397349437)

[Capa de Presentación 33](#_Toc397349438)

[Capa de Negocio 33](#_Toc397349439)

[Capa de Datos 35](#_Toc397349440)

[Capas y Niveles 35](#_Toc397349441)

[1.5 Desarrollo de la Aplicación 37](#_Toc397349442)

[1.5.1 Interfaz Gráfica de Usuario 37](#_Toc397349443)

[1.5.2 Hardware Empleado en la Implementación 44](#_Toc397349444)

[1.5.2.1 Lector de Código de Barras Symbol Ds 6608 2d 44](#_Toc397349445)

[1.5.2.2 Arduino Mega Rev 3 2560 46](#_Toc397349446)

[1.5.2.3 Sensor de Temperatura y Humedad DHT 22 48](#_Toc397349447)

[1.5.2.4 Cámara IP AXIS M1054 50](#_Toc397349448)

[1.5.3 Diagrama de Montaje de Componentes Electrónicos 52](#_Toc397349449)

[1.5.4 Lista de Componentes Electrónicos 54](#_Toc397349450)

[1.5.5 Código Fuente del Hardware Arduino Mega 2560 Rev 3 55](#_Toc397349451)

[Conclusiones 57](#_Toc397349452)

[Sugerencias 58](#_Toc397349453)

[Bibliografía 59](#_Toc397349454)

# Introducción

En el presente trabajo se pretende dar una solución eficaz a problemas que enfrenta las empresas del sector privado por hurto de combustible, para ello se implementara una aplicación que tendrá la función de: controlar el acceso de personal, controlar el abastecimiento de combustible mediante captura de (foto y video) y monitorear mediante sensores el estado del depósito de combustible para prevenir cualquier incidente.

Para realizar la aplicación se desarrollara utilizando: Oracle como gestor de la base de datos, C Sharp como lenguaje para la implementación de la interfaz de usuario, Arduino para la programación del hardware y finalmente para la captura de (Foto y Video) se utilizara la SDK AXISMediaControlSDK\_6\_32.

# Lista de figuras

[Figura 1 Diagrama de la Capa de Datos 11](#_Toc397347298)

[Figura 2 Diagrama de la Capa Entidad 12](#_Toc397347299)

[Figura 3 Diagrama de la Capa de Negocio 13](#_Toc397347300)

[Figura 4 Diagrama de la UI 14](#_Toc397347301)

[Figura 5 Diagrama de la UI 15](#_Toc397347302)

[Figura 6 Diagrama de la UI 16](#_Toc397347303)

[Figura 7 Diagrama de la UI 17](#_Toc397347304)

[Figura 8 Modelo típico de tres capas. 32](#_Toc397347305)

[Figura 9 Arquitectura en 4 Capas 33](#_Toc397347306)

[Figura 10 Ventana Principal IDSTORE 37](#_Toc397347307)

[Figura 11 Ventana de Ingreso de Nuevo Colaborador 38](#_Toc397347308)

[Figura 12 Ventana de Autorización de Acceso 38](#_Toc397347309)

[Figura 13 Ventana de Registro de Colaborador 39](#_Toc397347310)

[Figura 14 Ventana Para Asignar Código de Abastecimiento 40](#_Toc397347311)

[Figura 15 Ventana Para Abastecimiento de Combustible 40](#_Toc397347312)

[Figura 16 Ventana Para el Ingreso de Nuevas Areas 41](#_Toc397347313)

[Figura 17 Ventana para el Ingreso de Nuevos Cargos 41](#_Toc397347314)

[Figura 18 Ventana de Mantenimiento de Cámara Axis 42](#_Toc397347315)

[Figura 19 Ventana de Mantenimiento de Tanque 42](#_Toc397347316)

[Figura 20 Ventana de Monitoreo de Sensor de Temperatura y Humedad 43](#_Toc397347317)

[Figura 21 Lector de Código de Barras Symbol Ds 6608 2d. 45](#_Toc397347318)

[Figura 22 Arduino Mega Rev 3 2560. 47](#_Toc397347319)

[Figura 23 Sensor de Humedad y Temperatura DHT 22 49](#_Toc397347320)

[Figura 24 Esquema Fisicco Axis M1054 51](#_Toc397347321)

[Figura 25 Cámara IP AXIS M1054 51](#_Toc397347322)

[Figura 26 Diagrama de componentes electrónicos. 52](#_Toc397347323)

[Figura 27 Foto de Conexiones Finales 53](#_Toc397347324)

[Figura 28 Vista de valores de componentes electrónicos. 54](#_Toc397347325)

# 1.1 Descripción del Problema

La empresa Bouby SAC es una empresa que tiene como actividad principal la construcción y mantenimiento de carreteras, para realizar esta actividad la empresa requiere de grandes cantidades de combustible que deben ser almacenados en campo, para distribuir a los equipos.

La empresa controla la salida de distribución de combustible a los diferentes equipos pesados que realizaran labores de la actividad principal de la empresa, mediante el formato F\_1524 (ISO 9000).

La empresa ha registrado mayor consumo de combustible durante los últimos meses, esto hace sospechar que existe personal que está hurtando combustible en el área de trabajo, ya que el formato F\_1524 solo es un registro para el ingreso y salida de combustible, pero no registra realmente el momento que se realiza la distribución al equipo, también se presenta el problema del stock actual de combustible, para ello la empresa requiere un software que tenga un registro de la distribución de combustible en el que se debe registrar mediante video y foto el momento de la distribución, además este debe tener un registro del stock actual de combustible y como de seguridad debe controlar mediante sensores la temperatura del depósito para la acción inmediata en caso de incendio.

# 1.3 Obtención de los Requerimientos

## 1.3.1 Requerimientos Funcionales.

* El software debe permitir registrar los datos de los colaboradores e imprimir un fotocheck para poder acceder a las instalaciones de la empresa, además el acceso al depósito de combustible debe estar autorizado mediante un código.
* El software debe contar con un usuario administrador que permita otorgar el código de acceso al depósito de combustible, para que el colaborador pueda abastecerse de combustible los vehículos de la empresa.
* El abastecimiento de combustible debe poder realizarse únicamente mediante un código otorgado por el administrador en el que se podrá asignar de acuerdo al código de abastecimiento la cantidad de galones autorizados al colaborador.
* El software debe permitir la búsqueda de ingresos y salida del depósito de combustible por colaborador, además de realizar la búsqueda debe permitir la reproducción de video de acuerdo al criterio de búsqueda.
* El sistema debe cumplir las medidas de seguridad de las instalaciones, es decir no puede haber una distancia menor a 50m desde las instalaciones del servidor.
* El sistema debe emitir una alarma en caso de incremento de temperatura o humedad en el depósito de combustible.

## 1.3.2 Requerimientos no Funcionales.

* La búsqueda de información no debe tardar más de cinco segundos.
* Si se supera este plazo, el sistema detiene la búsqueda y muestra los resultados encontrados.
* La captura de los sensores no debe tardar más de 2 segundos.

# 1.4 Arquitectura del Software

## 1.4.1 Diagramas de Clases

El diagrama de clases es un tipo de diagrama estático que describe la estructura del sistema mostrando sus clases, orientados a objetos.

En el presente trabajo se aprecia las clases de las capas del software., esto nos permitirá apreciar los diferentes métodos, funciones y atributos que se implementara en la aplicación

### 1.4.1.1 Diagrama de Clases de la Capa de Datos

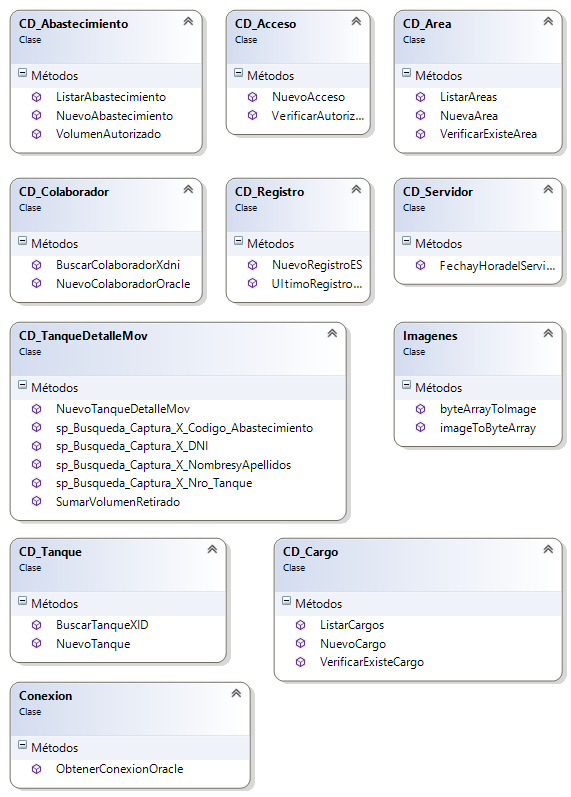


Figura 1 Diagrama de la Capa de Datos

### 1.4.1.2 Diagrama de Clases de la Capa Entidad

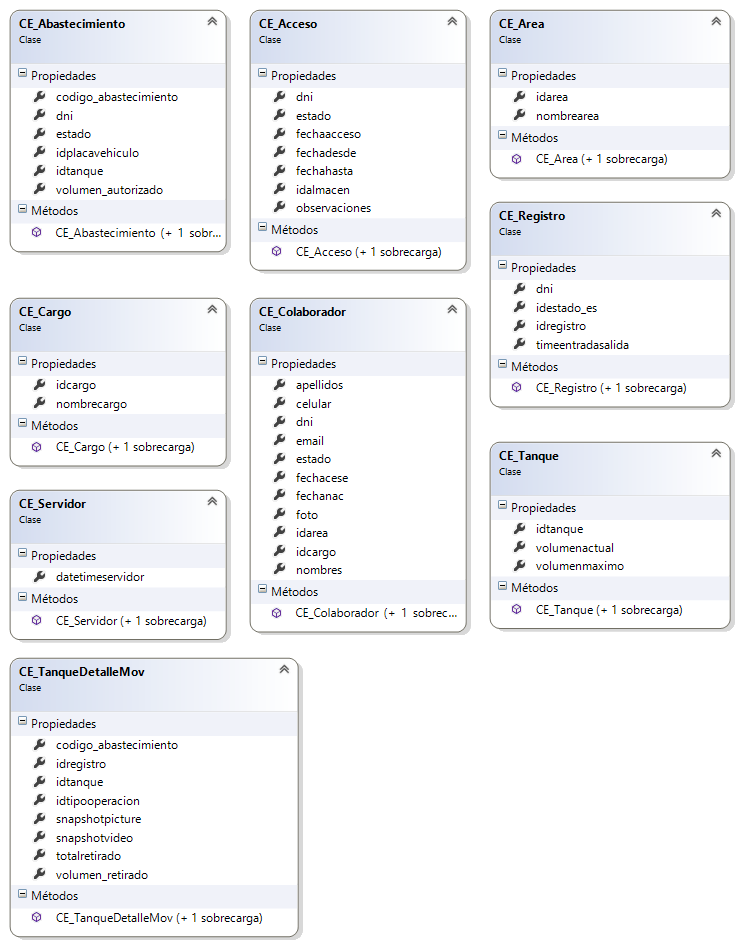


Figura 2 Diagrama de la Capa Entidad

### 1.4.1.3 Diagrama de Clases de la Capa de Negocio



Figura 3 Diagrama de la Capa de Negocio

### 1.4.1.4 Diagrama de Clases de la Capa UI

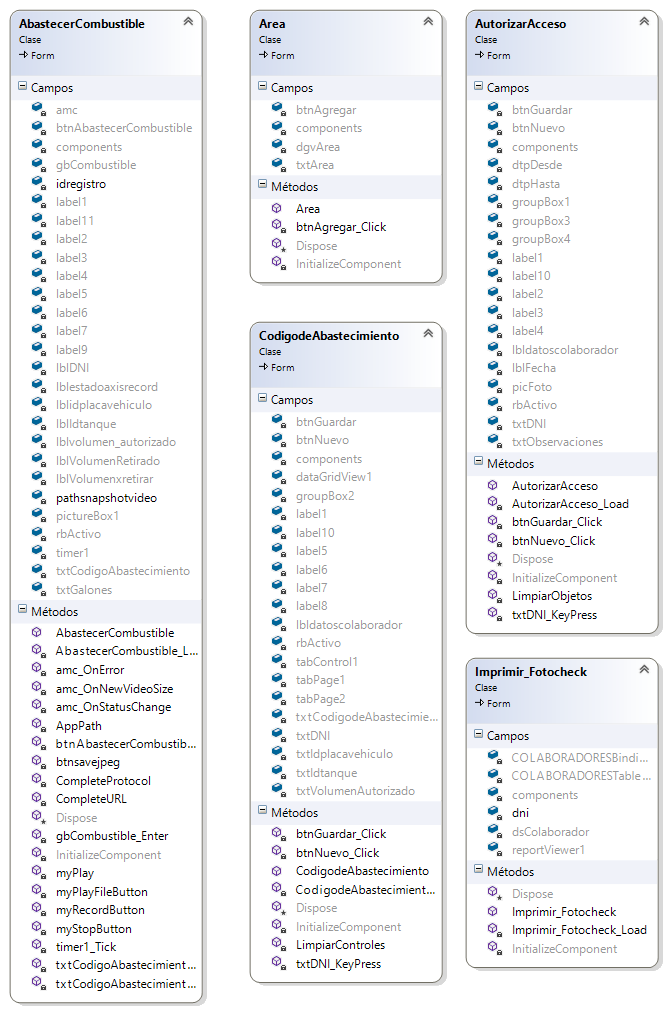


Figura 4 Diagrama de la UI

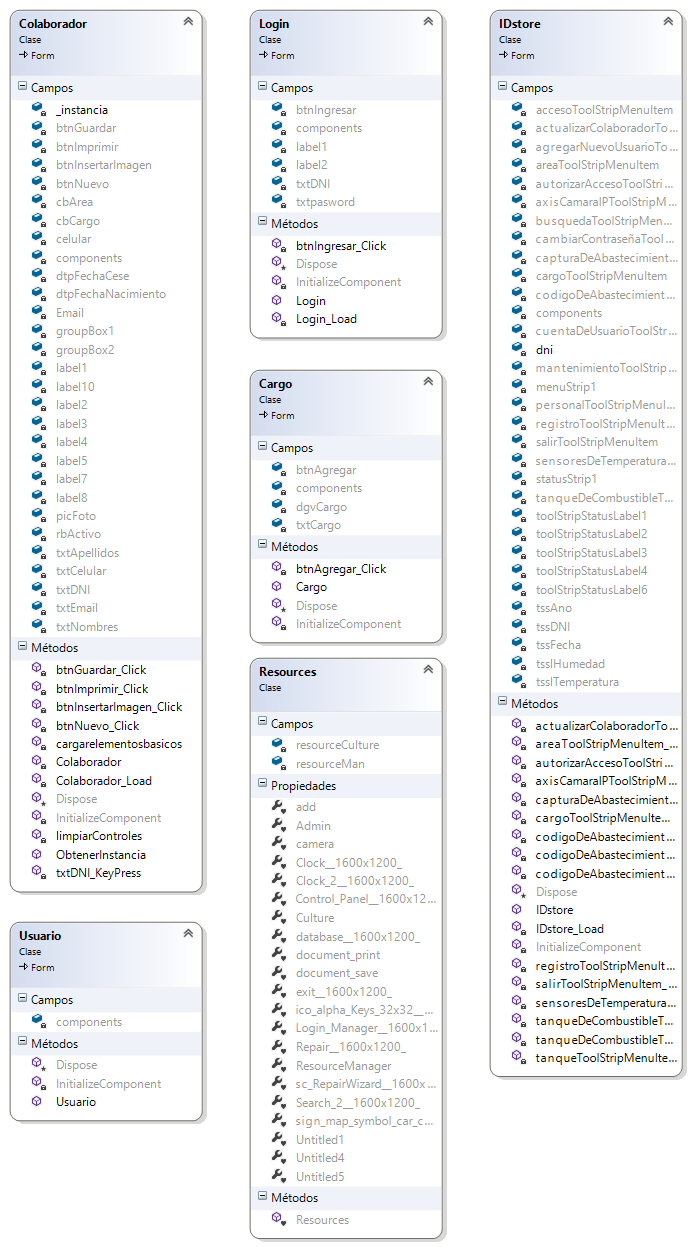


Figura 5 Diagrama de la UI

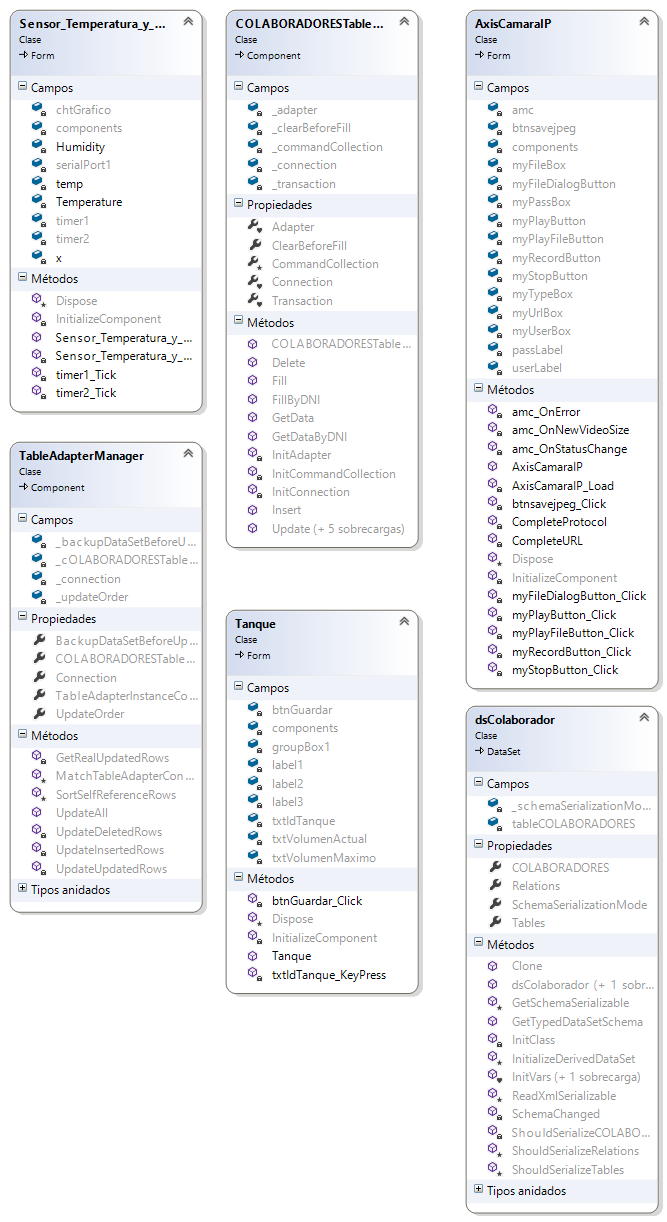


Figura 6 Diagrama de la UI

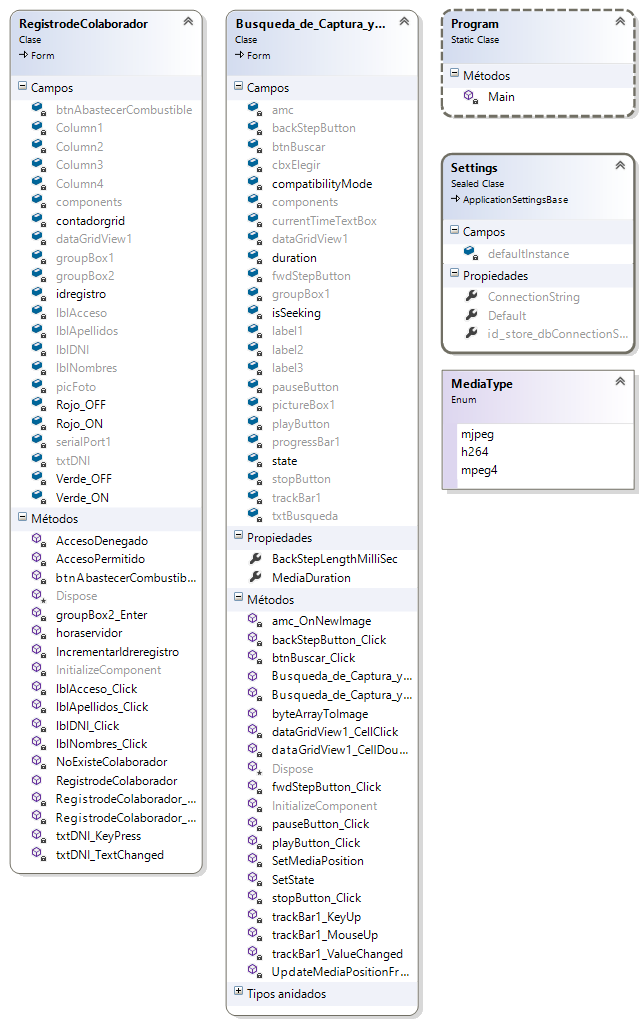


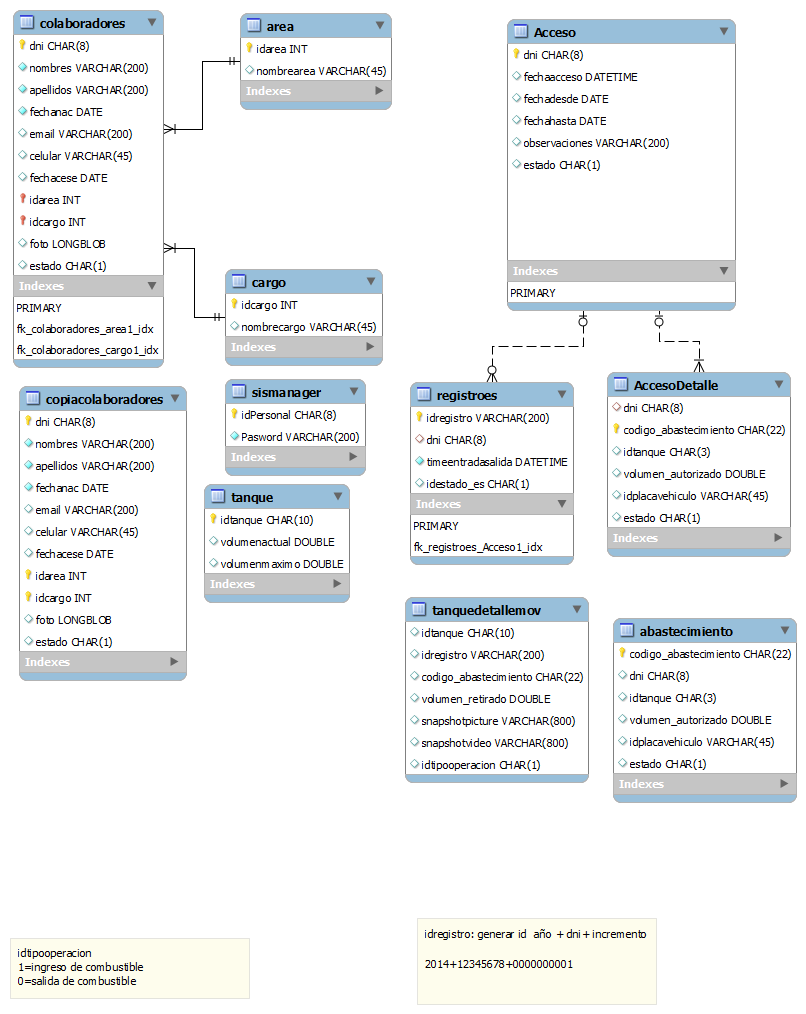
Figura 7 Diagrama de la UI

## 1.4.2 Diagramas de la Base de Datos

Un diagrama o modelo entidad-relación (a veces denominado por sus siglas en inglés, E-R "Entity relationship", o del español DER "Diagrama de Entidad Relación") es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades.

### 1.4.2.1 Diagrama Lógico de la Base de datos

El diagrama físico final será implementado de acuerdo al diagrama lógico siguiente:



### 1.4.2.2 Tablas

Las tablas de una base de datos conforman la estructura principal de la misma, podríamos decir que se trata de los cimientos de nuestra estructura. Cada tabla contiene la información que utilizarán el resto de objetos de la base de datos.

Las tablas implementadas son las siguientes:

La tabla ABASTECIMIENTO es creada para almacenar el código de abastecimiento, así como también todos los detalles del operador y vehículo autorizado.

CREATE TABLE "ABASTECIMIENTO"

( "CODIGO\_ABASTECIMIENTO" VARCHAR2(22),

"DNI" VARCHAR2(8),

"IDTANQUE" VARCHAR2(3),

"VOLUMEN\_AUTORIZADO" NUMBER(8,2),

"IDPLACAVEHICULO" VARCHAR2(45),

"ESTADO" VARCHAR2(1)

) ;

La tabla ACCESO nos permitirá almacenar de acuerdo al número de DNI el acceso autorizado mediante un intervalo de fechas otorgados por el administrador del software.

CREATE TABLE "ACCESO"

( "DNI" VARCHAR2(8),

"FECHAACCESO" TIMESTAMP (6),

"FECHADESDE" DATE,

"FECHAHASTA" DATE,

"OBSERVACIONES" VARCHAR2(200),

"ESTADO" VARCHAR2(1),

PRIMARY KEY ("DNI") ENABLE

) ;

La entidad AREA servirá para registrar las áreas de la empresa.

CREATE TABLE "AREA"

( "IDAREA" NUMBER(3,0),

"NOMBREAREA" VARCHAR2(45),

PRIMARY KEY ("IDAREA") ENABLE

) ;

La entidad CARGO servirá para registrar las ocupaciones de los colaboradores en la empresa.

CREATE TABLE "CARGO"

( "IDCARGO" NUMBER(3,0),

"NOMBRECARGO" VARCHAR2(45),

PRIMARY KEY ("IDCARGO") ENABLE

) ;

La tabla COLABORADORES será utilizada para almacenar la información de los colaboradores.

CREATE TABLE "COLABORADORES"

( "DNI" VARCHAR2(8),

"NOMBRES" VARCHAR2(200),

"APELLIDOS" VARCHAR2(200),

"FECHANAC" DATE,

"EMAIL" VARCHAR2(200),

"CELULAR" VARCHAR2(45),

"FECHACESE" DATE,

"IDAREA" NUMBER(3,0),

"IDCARGO" NUMBER(3,0),

"FOTO" BLOB,

"ESTADO" VARCHAR2(1),

PRIMARY KEY ("DNI") ENABLE

) ;

La tabla COPIACOLABORADORES será utilizada para almacenar la información de los colaboradores, pero como copia de seguridad.

CREATE TABLE "COPIACOLABORADORES"

( "DNI" VARCHAR2(8),

"NOMBRES" VARCHAR2(200),

"APELLIDOS" VARCHAR2(200),

"FECHANAC" DATE,

"EMAIL" VARCHAR2(200),

"CELULAR" VARCHAR2(45),

"FECHACESE" DATE,

"IDAREA" NUMBER(3,0),

"IDCARGO" NUMBER(3,0),

"FOTO" BLOB,

"ESTADO" VARCHAR2(1),

PRIMARY KEY ("DNI") ENABLE

) ;

Esta tabla REGISTROES servirá para registrar todos los ingresos y salidas del personal.

CREATE TABLE "REGISTROES"

( "IDREGISTRO" VARCHAR2(200),

"DNI" VARCHAR2(8),

"TIMEENTRADASALIDA" TIMESTAMP (6),

"IDESTADO\_ES" VARCHAR2(1)

) ;

El objeto SISMANAGER será de repositorio para almacenar los usuarios con rol administrador.

CREATE TABLE "SISMANAGER"

( "IDPERSONAL" VARCHAR2(8),

"PASWORD" VARCHAR2(200),

PRIMARY KEY ("IDPERSONAL") ENABLE

) ;

La entidad TANQUE permite tener un registro de los tanques de combustible de la empresa.

CREATE TABLE "TANQUE"

( "IDTANQUE" VARCHAR2(10),

"VOLUMENACTUAL" NUMBER(8,2),

"VOLUMENMAXIMO" NUMBER(8,2),

PRIMARY KEY ("IDTANQUE") ENABLE

) ;

La tabla TANQUEDETALLEMOV será creada únicamente para registrar el detalle de movimiento de combustible.

CREATE TABLE "TANQUEDETALLEMOV"

( "IDTANQUE" VARCHAR2(10),

"IDREGISTRO" VARCHAR2(200),

"CODIGO\_ABASTECIMIENTO" VARCHAR2(22),

"VOLUMEN\_RETIRADO" NUMBER(8,0),

"SNAPSHOTPICTURE" VARCHAR2(800),

"SNAPSHOTVIDEO" VARCHAR2(800),

"IDTIPOOPERACION" VARCHAR2(1)

) ;

### 1.4.2.3 Secuencias

ORACLE proporciona los objetos de secuencia para la generación de códigos numéricos automáticos.

Las secuencias son una solución fácil y elegante al problema de los códigos autogenerados.

Las secuencias implementadas son las siguientes:

La secuencia AUTOINCREMENTOAREA servirá exclusivamente para autogenerar el id para el campo idarea de la tabla AREA.

CREATE SEQUENCE "AUTOINCREMENTOAREA" MINVALUE 1 MAXVALUE 1000 INCREMENT BY 1 START WITH 3 NOCACHE NOORDER NOCYCLE ;

La secuencia AUTOINCREMENTOCARGO servirá exclusivamente para autogenerar el id para el campo idcargo de la tabla CARGO.

CREATE SEQUENCE "AUTOINCREMENTOCARGO" MINVALUE 1 MAXVALUE 1000 INCREMENT BY 1 START WITH 2 NOCACHE NOORDER NOCYCLE ;

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

### 1.4.2.4 Índices

Los índices sirven para acceder a los registros de una tabla rápidamente, acelerando la localización de la información.

Un índice posibilita el acceso directo y rápido haciendo más eficiente las búsquedas. Sin índice, Oracle debe recorrer secuencialmente toda la tabla para encontrar un registro.

Los índices implementados son los siguientes:

CREATE INDEX "ABASTECIMIENTO\_CTX1" ON "ABASTECIMIENTO" ("CODIGO\_ABASTECIMIENTO")

INDEXTYPE IS "CTXSYS"."CONTEXT" ;

CREATE INDEX "ACCESO\_CTX1" ON "ACCESO" ("DNI")

INDEXTYPE IS "CTXSYS"."CONTEXT" ;

CREATE INDEX "AREA\_CTX1" ON "AREA" ("NOMBREAREA")

INDEXTYPE IS "CTXSYS"."CONTEXT" ;

CREATE INDEX "CARGO\_CTX1" ON "CARGO" ("NOMBRECARGO")

INDEXTYPE IS "CTXSYS"."CONTEXT" ;

CREATE INDEX "COLABORADORES\_CTX1" ON "COLABORADORES" ("DNI")

INDEXTYPE IS "CTXSYS"."CONTEXT" ;

CREATE INDEX "REGISTROES\_CTX1" ON "REGISTROES" ("IDREGISTRO")

INDEXTYPE IS "CTXSYS"."CONTEXT" ;

CREATE INDEX "TANQUEDETALLEMOV\_CTX1" ON "TANQUEDETALLEMOV" ("IDTANQUE")

INDEXTYPE IS "CTXSYS"."CONTEXT" ;

### 1.4.2.5 Vistas

Una vista es un objeto. Una vista es una alternativa para mostrar datos de varias tablas; es como una tabla virtual que almacena una consulta. Los datos accesibles a través de la vista no están almacenados en la base de datos, en la base de datos se guarda la definición de la vista y no el resultado de ella.

Las vistas permiten:

- simplificar la administración de los permisos de usuario: se pueden dar al usuario permisos para que solamente pueda acceder a los datos a través de vistas, en lugar de concederle permisos para acceder a ciertos campos, así se protegen las tablas base de cambios en su estructura.

- mejorar el rendimiento: se puede evitar escribir instrucciones repetidamente almacenando en una vista el resultado de una consulta compleja que incluya información de varias tablas.

Las vistas implementadas son las siguientes:

La vista LISTAR\_AREA permitirá recuperar todas las áreas de la empresa.

CREATE OR REPLACE FORCE VIEW "LISTAR\_AREA" ("IDAREA", "NOMBREAREA") AS

select "IDAREA","NOMBREAREA" from area;

La vista LISTAR\_CARGO permitirá recuperar todos los cargos de la empresa.

CREATE OR REPLACE FORCE VIEW "LISTAR\_CARGO" ("IDCARGO", "NOMBRECARGO") AS

select "IDCARGO","NOMBRECARGO" from cargo;

### 1.4.2.6 Procedimientos Almacenados

Un procedimiento es un subprograma que ejecuta una acción específica y que no devuelve ningún valor. Un procedimiento tiene un nombre, un conjunto de parámetros (opcional) y un bloque de código.

Los procedimientos almacenados implementados son los siguientes:

El procedimiento almacenado SP\_NUEVA\_AREA nos permite ingresar nuevas áreas.

create or replace procedure "SP\_NUEVA\_AREA"  
(nombreareax IN VARCHAR2)  
is  
begin  
INSERT INTO area(IDAREA,nombrearea) VALUES(autoincrementoarea.NextVal,nombreareax);  
end SP\_NUEVA\_AREA;​

El procedimiento almacenado sp\_NuevoAcceso permite que podamos registrar un nuevo acceso de acuerdo al dni y en un intervalo de fechas.

create or replace PROCEDURE "sp\_NuevoAcceso" ( dni in  varchar2,  
     fechaacceso  timestamp,  
     fechadesde  timestamp,  
     fechahasta  timestamp,  
     observaciones  varchar2,  
     estado  varchar2  
    )  
    is  
BEGIN  
INSERT INTO acceso  
(dni,fechaacceso,fechadesde,fechahasta,observaciones,estado)  
VALUES(dni,fechaacceso,fechadesde,fechahasta,observaciones,estado);  
END sp\_NuevoAcceso;​

El SP\_NUEVO\_ABASTECIMIENTO registra un nuevo código de abastecimiento, teniendo presente los datos del operador y el vehiculo, asi como también los galones autorizados.

create or replace procedure "SP\_NUEVO\_ABASTECIMIENTO"  
(codigo\_abastecimiento IN VARCHAR2 , dni IN VARCHAR2, idtanque IN VARCHAR2 , volumen\_autorizado IN NUMBER, idplacavehiculo IN VARCHAR2, estado IN VARCHAR2 )  
is  
begin  
INSERT INTO abastecimiento(codigo\_abastecimiento,dni,idtanque, volumen\_autorizado, idplacavehiculo, estado)   
VALUES(codigo\_abastecimiento,dni,idtanque,volumen\_autorizado,idplacavehiculo,estado);  
end SP\_NUEVO\_ABASTECIMIENTO;​

El procedimiento almacenado SP\_NUEVO\_CARGO nos permite ingresar un nuevo cargo.

create or replace procedure "SP\_NUEVO\_CARGO"  
(nombrecargox IN VARCHAR2)  
is  
begin  
INSERT INTO cargo(IDCARGO, nombrecargo) VALUES(autoincrementocargo.NextVal,nombrecargox);  
end SP\_NUEVO\_CARGO;​

create or replace procedure sp\_Nuevo\_Colaborador (dni  in varchar2,  
     nombres  in varchar2,  
     apellidos  in varchar2,  
     fechanac  in date,  
     email  in varchar2,  
     celular  in varchar2,  
     fechacese  in date,  
     idarea  in number,  
     idcargo  in number,  
     foto  in blob,  
    estado in varchar2)  
is  
BEGIN  
 INSERT INTO colaboradores  
(dni,nombres,apellidos,fechanac,email,celular,fechacese,idarea,idcargo,foto,estado)  
VALUES(dni,nombres,apellidos,fechanac,email,celular,fechacese,idarea,  
idcargo,foto,estado);  
END sp\_Nuevo\_Colaborador;​

El sp\_nuevo\_registro\_es registrara el ingreso y salida de personal a las instalaciones de la empresa.

create or replace procedure “sp\_nuevo\_registro\_es”  
(dni IN VARCHAR2,idregistro IN VARCHAR2,  idestado\_es IN VARCHAR2)  
is  
begin  
INSERT INTO registroes (idregistro, dni,timeentradasalida,idestado\_es) VALUES (idregistro, dni, current\_timestamp, idestado\_es);  
end sp\_nuevo\_registro\_es;​

create or replace PROCEDURE sp\_Nuevo\_Tanque(idtanque in varchar2, volumenactual in number, volumenmaximo in number)  
is  
BEGIN  
INSERT INTO tanque  
(idtanque,  
volumenactual,  
volumenmaximo)  
VALUES  
(idtanque,  
volumenactual,  
volumenmaximo);  
END  sp\_Nuevo\_Tanque;​

En este procedimiento SP\_NUEVO\_TANQUE\_DETALLE\_MOV podremos registrar los movimientos de combustible que realiza los operadores, asi como también la captura en el momento exacto que lo realiza.

create or replace procedure "SP\_NUEVO\_TANQUE\_DETALLE\_MOV"  
(idtanque IN VARCHAR2,  
idregistro IN VARCHAR2,  
codigo\_abastecimiento IN VARCHAR2,  
volumen\_retirado IN NUMBER,  
snapshotpicture IN VARCHAR2,  
snapshotvideo IN VARCHAR2,  
idtipooperacion IN VARCHAR2)  
is  
begin  
INSERT INTO tanquedetallemov (idtanque,idregistro, codigo\_abastecimiento,volumen\_retirado, snapshotpicture, snapshotvideo, idtipooperacion) VALUES (idtanque, idregistro,  
codigo\_abastecimiento, volumen\_retirado, snapshotpicture, snapshotvideo, idtipooperacion);  
end;​

### 1.4.2.7 Desencadenadores

Un "trigger" (disparador o desencadenador) es un bloque de código que se ejecuta automáticamente cuando ocurre algún evento (como inserción, actualización o borrado) sobre una determinada tabla (o vista); es decir, cuando se intenta modificar los datos de una tabla (o vista) asociada al disparador.

Se crean para conservar la integridad referencial y la coherencia entre los datos entre distintas tablas; para registrar los cambios que se efectúan sobre las tablas y la identidad de quien los realizó; para realizar cualquier acción cuando una tabla es modificada; etc.

Las desencadenadores implementados son los siguientes:

create or replace TRIGGER  abastecimiento\_no\_borrar  
BEFORE

delete on abastecimiento  
for each row  
begin  
RAISE\_APPLICATION\_ERROR (-20500,'No esta permitido borrar registros de la tabla abastecimiento');  
  
end abastecimiento\_no\_borrar;​

create or replace TRIGGER  acceso\_no\_borrar  
BEFORE  
delete on acceso  
for each row  
begin  
RAISE\_APPLICATION\_ERROR (-20500,'No esta permitido borrar registros de la tabla acceso');  
  
end acceso\_no\_borrar;​

create or replace trigger "BACKUP\_COLABORADORES"  
AFTER  
insert or update on "COLABORADORES"  
for each row  
begin  
IF INSERTING THEN  
  
 INSERT INTO copiacolaboradores(dni,  
nombres,  
apellidos,  
fechanac,  
email,  
celular,  
fechacese,  
idarea,  
idcargo,  
foto,  
estado)  
VALUES  
(:new.dni,  
:new.nombres,  
:new.apellidos,  
:new.fechanac,  
:new.email,  
:new.celular,  
:new.fechacese,  
:new.idarea,  
:new.idcargo,  
:new.foto,  
:new.estado);  
  
ELSIF UPDATING  THEN  
  
update copiacolaboradores set   
dni=:new.dni,  
nombres=:new.nombres,  
apellidos=:new.apellidos,  
fechanac=:new.fechanac,  
email=:new.email,  
celular=:new.celular,  
fechacese=:new.fechacese,  
idarea=:new.idarea,  
idcargo=:new.idcargo,  
foto=:new.foto,  
estado=:new.estado  
where dni= :old.dni;  
  
END IF;  
end;​

create or replace TRIGGER  colaboradores\_no\_borrar  
BEFORE  
delete on colaboradores  
for each row  
begin  
RAISE\_APPLICATION\_ERROR (-20500,'No esta permitido borrar registros de la tabla colaboradores');  
  
end;​

create or replace TRIGGER  registroes\_no\_borrar  
BEFORE  
delete on registroes  
for each row  
begin  
RAISE\_APPLICATION\_ERROR (-20500,'No esta permitido borrar registros de la tabla registroes');  
  
end registroes\_no\_borrar;​

create or replace trigger "SEGURIDAD\_COLABORADORES"  
BEFORE  
insert on "COLABORADORES"  
for each row  
begin  
IF (TO\_CHAR (sysdate,'DY') IN ('SAT','SUN')) OR (TO\_CHAR(sysdate,'HH24')NOT BETWEEN  '08' AND '16') THEN   
RAISE\_APPLICATION\_ERROR (-20500,  'Ud solo puede insertar colaboradores durante  horarios de trabajo');  
END IF;  
  
end;​

create or replace TRIGGER  tanquedetallemov\_no\_borrar  
BEFORE  
delete on tanquedetallemov  
for each row  
begin  
RAISE\_APPLICATION\_ERROR (-20500,'No esta permitido borrar registros de la tabla tanque detalle');  
  
end tanquedetallemov\_no\_borrar;​

create or replace TRIGGER  tanque\_no\_borrar  
BEFORE  
delete on tanque  
for each row  
begin  
RAISE\_APPLICATION\_ERROR (-20500,'No esta permitido borrar registros de la tabla tanque');  
  
end tanque\_no\_borrar;​

## 1.4.3 Patrón de Diseño

La realización de Sistemas de Información se ha venido desarrollando en base a técnicas de programación, principalmente; la programación estructurada, luego en combinación utilizando la programación por eventos, actualmente se pudiera decir que se ha llegado a una madurez con la potencialidad de la programación orientada a objetos por la ventaja en la reutilización de código. En adición a ellas, se cuenta actualmente con la programación en n capas que hace uso de la programación orientada a objetos; la cual consiste en separar el código fuente según el rol, responsabilidad y funcionalidad; por ende el desarrollo es más rápido, y resulta más fácil el darle mantenimiento al Sistema.

### 1.4.3.1 **Programación en N Capas**

El estilo arquitectural en n capas se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de la interacción con otras capas y las responsabilidades la funcionalidad que implementan.

Cuanto más se aumenta el proceso operativo de la empresa, las necesidades de proceso crecen hasta desbordar las máquinas. Es por ello que se separa la estructura de un programa en varias capas.



Figura 8 Modelo típico de tres capas.

En adición a lo citado, podemos decir que actualmente la programación por capas es un estilo de programación en la que el objetivo principal es separar la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de negocios y ésta a su vez de la capa de presentación al usuario. El diseño que actualmente más se utiliza es el diseño en tres capas; sin embargo, la programación puede desglosarse en más capas, tal cual se presenta en la presente aplicación.

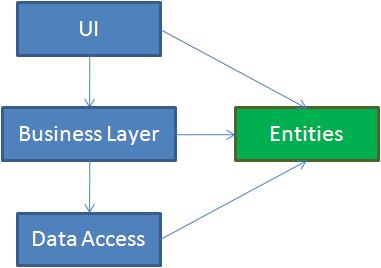


Figura 9 Arquitectura en 4 Capas

### Capa de Presentación

Es la responsable de la presentación visual de la aplicación. La capa de presentación enviará mensajes a los objetos de esta capa de negocios o intermedia, la cual o bien responderá entonces directamente o mantendrá un diálogo con la capa de la base de datos, la cual proporcionará los datos que se mandarían como respuesta a la capa de presentación.

Podemos decir que es la que se presenta al usuario, llamada también formulario o interfaz de presentación, esta captura los datos del usuario en el formulario e invoca a la capa de negocio, trasmitiéndole los requerimientos del usuario, ya sea de almacenaje, edición, o de recuperación de la información para la consulta respectiva.

### Capa de Negocio

Es la responsable del procesamiento que tiene lugar en la aplicación. Por ejemplo, en una aplicación bancaria el código de la capa de presentación se relacionaría simplemente con la monitorización de sucesos y con el envío de datos a la capa de procesamiento. Esta capa intermedia contendría los objetos que se corresponden con las entidades de la aplicación. Esta capa intermedia es la que conlleva capacidad de mantenimiento y de reutilización.

Contendrá objetos definidos por clases reutilizables que se pueden utilizar una y otra vez en otras aplicaciones. Estos objetos se suelen llamar objetos de negocios y son los que contienen la gama normal de constructores, métodos para establecer y obtener variables, métodos que llevan a cabo cálculos y métodos, normalmente privados, en comunicación con la capa de la base de datos.

Es en esta capa donde se reciben los requerimientos del usuario y se envían las respuestas tras el proceso, a requerimiento de la capa de presentación. Se denomina capa de negocio o lógica del negocio, es aquí donde se establecen todas las reglas que deben cumplirse. En realidad se puede tratar de varias funciones, por ejemplo, puede controlar la integridad referencial, otro que se encargue de la interfaz, tal como abrir y cerrar ciertos formularios o funcionalidades que tengan que ver con la seguridad, menús, etc., tiene los métodos que serán llamados desde las distintas partes de la interfaz o para acceder a la capa de datos, tal como se apreciará en la aplicación.

Esta capa interactúa con la capa de presentación para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al manejador de base de datos que realice una operación de almacenamiento, edición, eliminación, consulta de datos u otra.

### Capa de Datos

Esta capa se encarga de acceder a los datos, se debe usar la capa de datos para almacenar y recuperar toda la información de sincronización del Sistema. Es aquí donde se implementa las conexiones al servidor y la base de datos propiamente dicha, se invoca a los procedimientos almacenados los cuales reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. Todas estas capas pueden residir en un único ordenador (no debería ser lo usual), pero es lo más frecuente.

En sistemas complejos se llega a tener varios ordenadores sobre los cuales reside la capa de datos, y otra serie de ordenadores sobre los cuales reside la base de datos. Se recomienda que si el crecimiento de las necesidades o complejidad aumenta se debe separar en dos o más ordenadores, los cuales recibirán las peticiones del ordenador en que resida la capa de negocio. Esta recomendación es válida para la capa de negocios.

### Capas y Niveles

Es importante distinguir los conceptos de “Capas” (Layers) y “Niveles” (Tiers). Las capas se ocupan de la división lógica de componentes y funcionalidad y no tienen en cuenta la localización física de componentes en diferentes servidores o en diferentes lugares. Por el contrario, los Niveles se ocupan de la distribución física de componentes y funcionalidad en servidores separados. Teniendo en cuenta topología de redes y localizaciones remotas.

Las arquitecturas de N niveles facilitan la presencia de sistemas distribuidos en los que se pueden dividir los servicios y aumentar la escalabilidad y mantenimiento de los mismos.

# 1.5 Desarrollo de la Aplicación

## 1.5.1 Interfaz Gráfica de Usuario

La ventana principal de la aplicación con diseño MDI para un mejor control de las ventanas hijas.

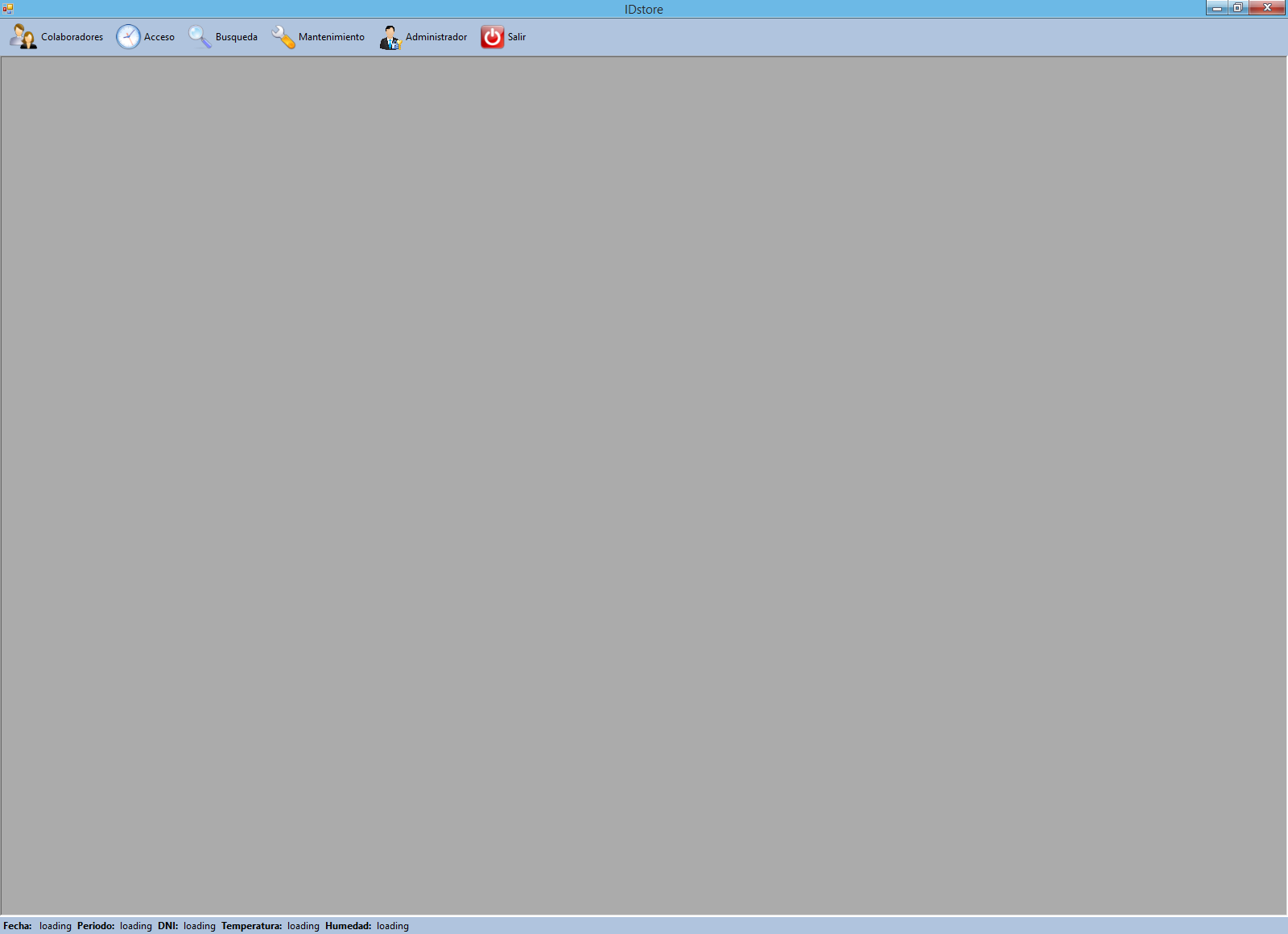


Figura 10 Ventana Principal IDSTORE

En este formulario el administrador del software podrá registra a los nuevos colaboradores que ingresen a trabajar.

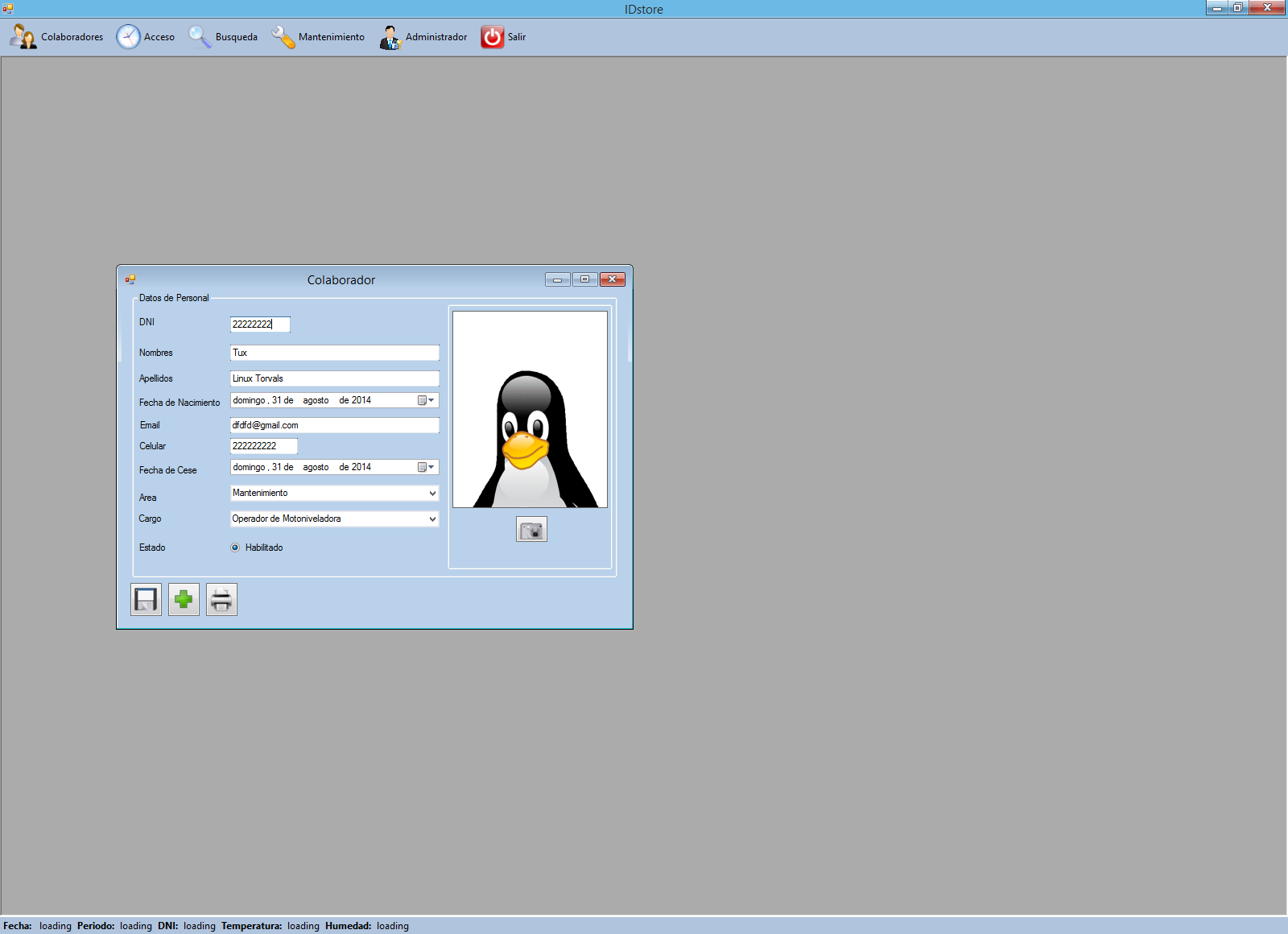


Figura 11 Ventana de Ingreso de Nuevo Colaborador

El formulario Autorizar Acceso servirá para poder autorizar el acceso al personal. La autorización de acceso será determinada de acuerdo a un intervalo de fecha.

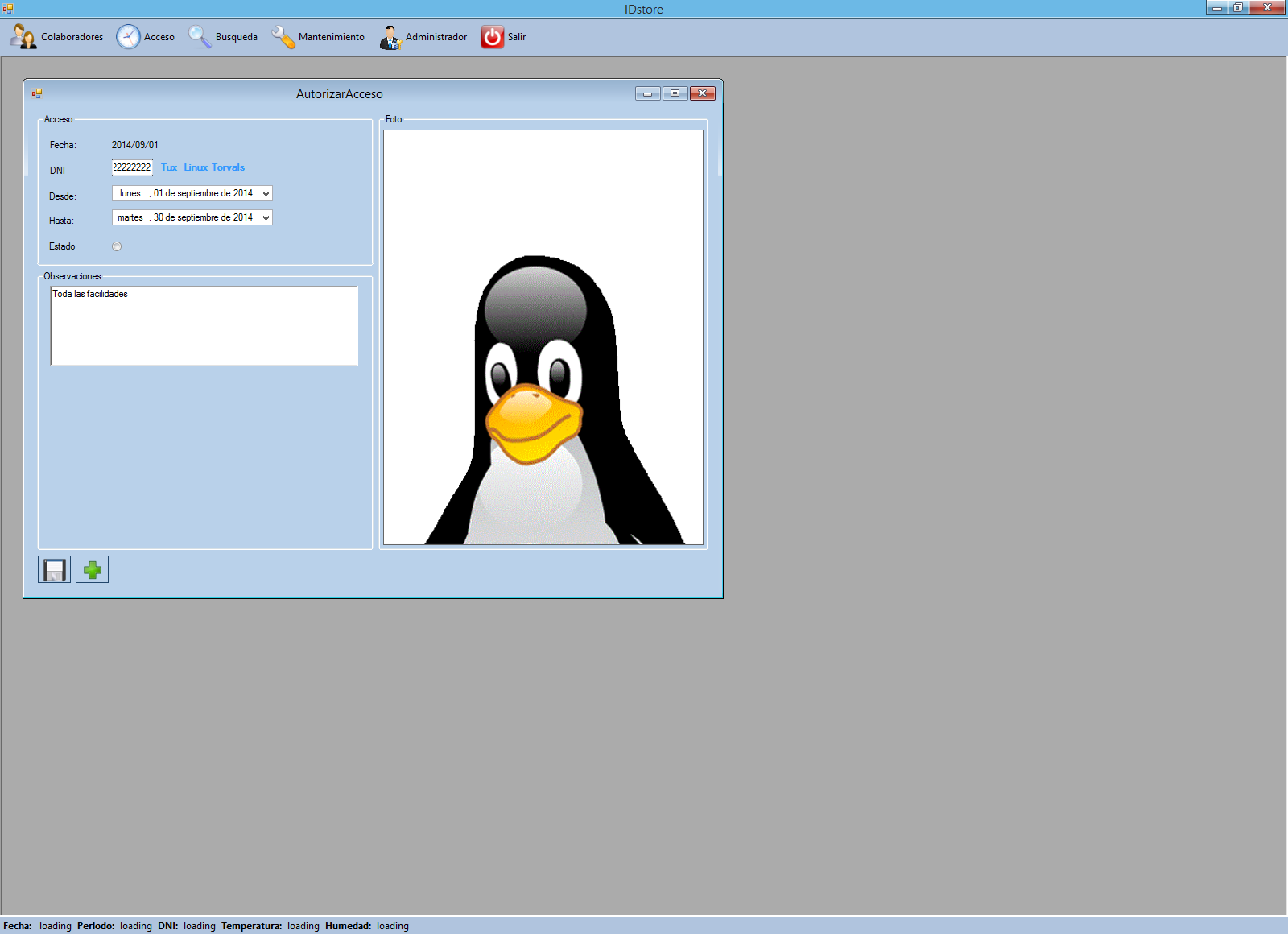


Figura 12 Ventana de Autorización de Acceso

Este formulario siempre permanecerá activa ya que permitirá a los colaboradores, poder identificarse mediante el lector de código de barras para poder acceder a las instalaciones de la empresa, así como también realizar el abastecimiento de combustible, siempre que tenga la autorización.

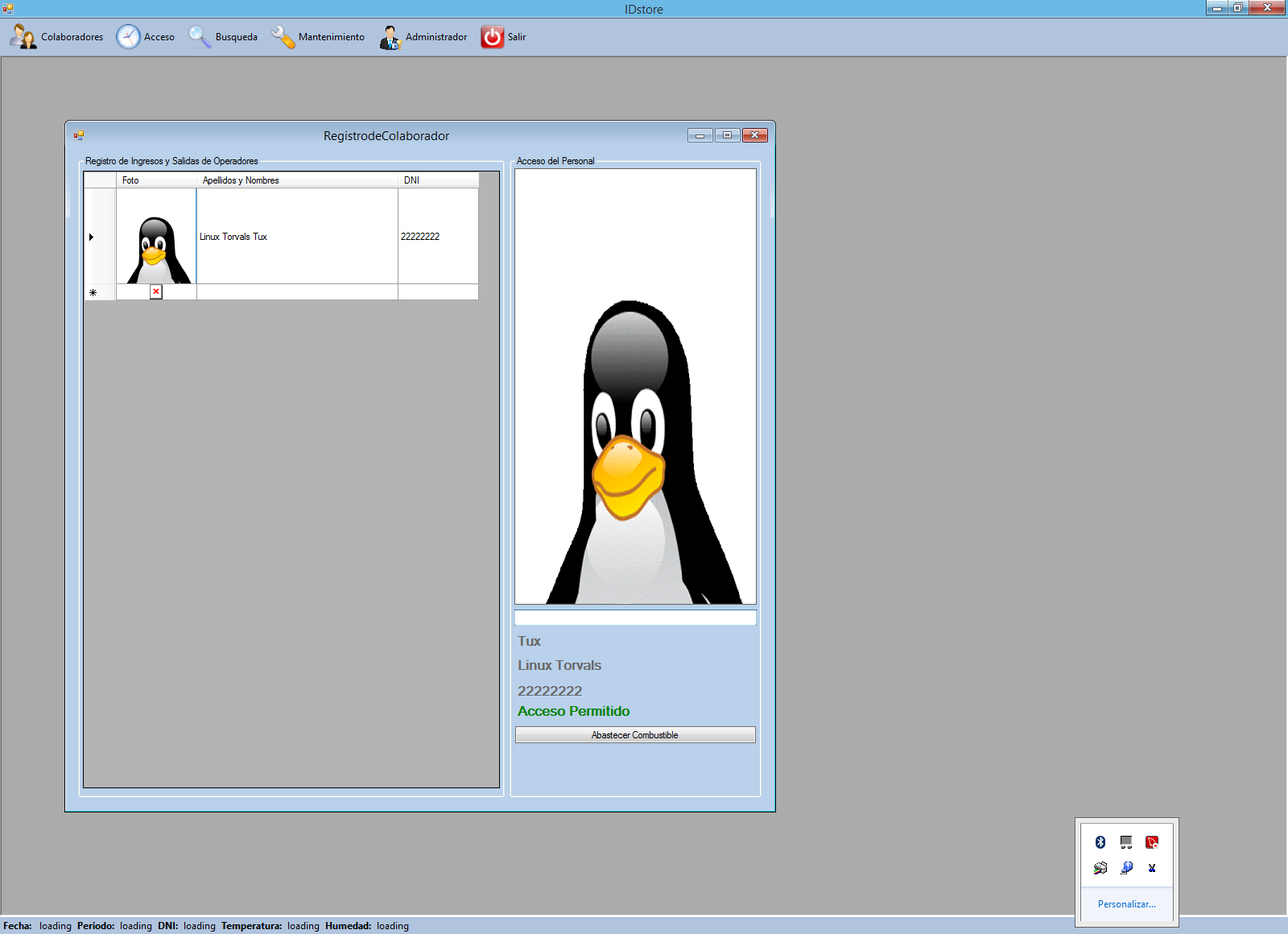


Figura 13 Ventana de Registro de Colaborador

El formulario Código Abastecimiento permite poder asignar un código de abastecimiento al personal.

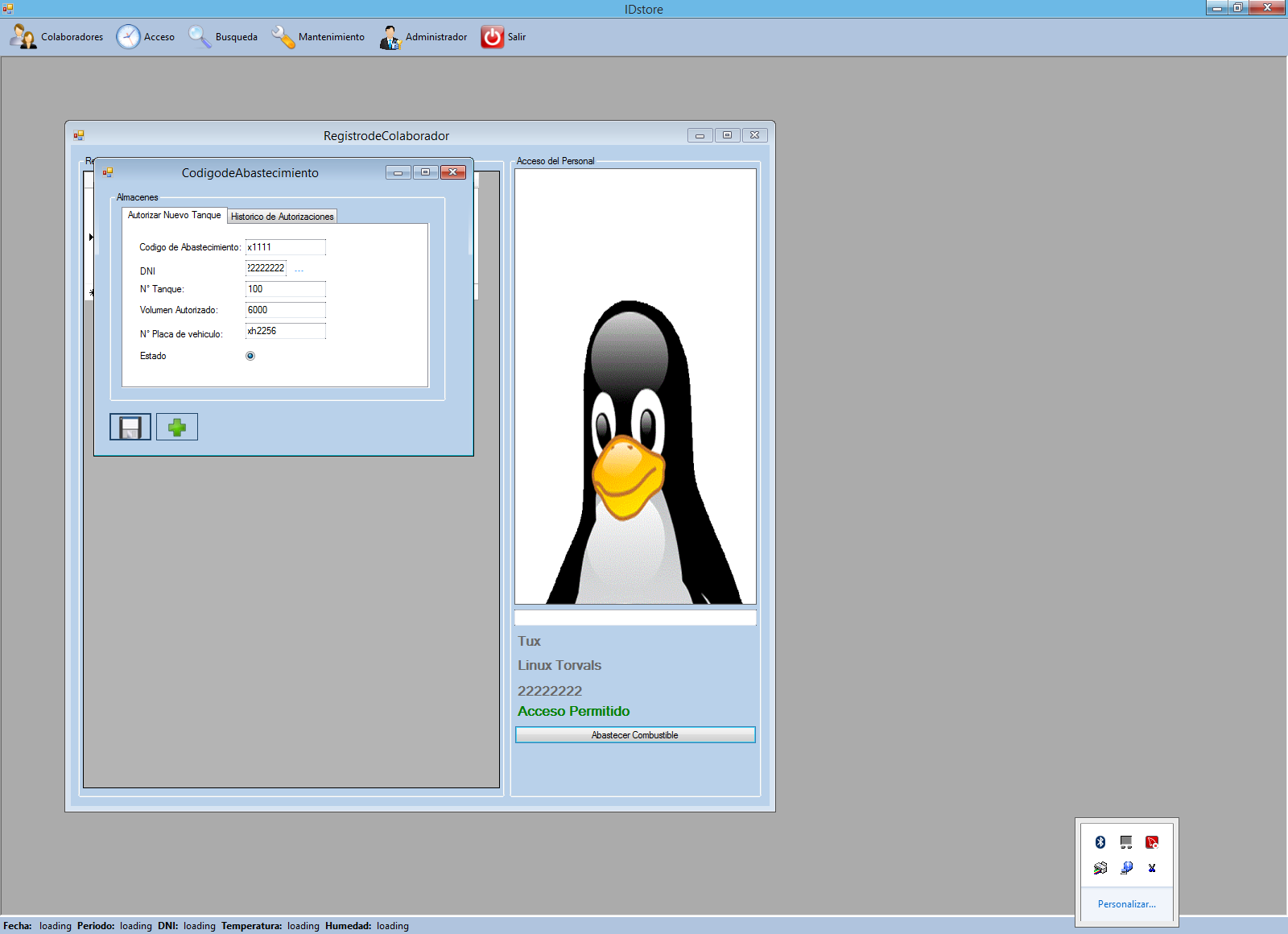


Figura 14 Ventana Para Asignar Código de Abastecimiento

A este formulario solamente se podrá acceder si se tiene el permiso y el código de abastecimiento correspondiente.

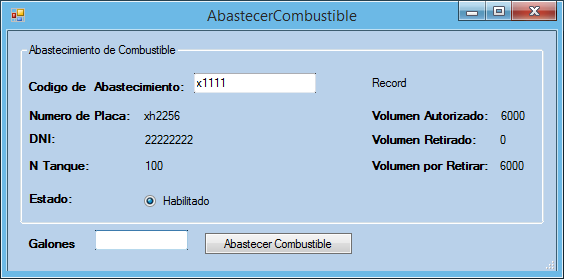


Figura 15 Ventana Para Abastecimiento de Combustible

Los siguientes formularios: Área y cargo permiten poder ingresar nuevas áreas y cargos de la empresa.

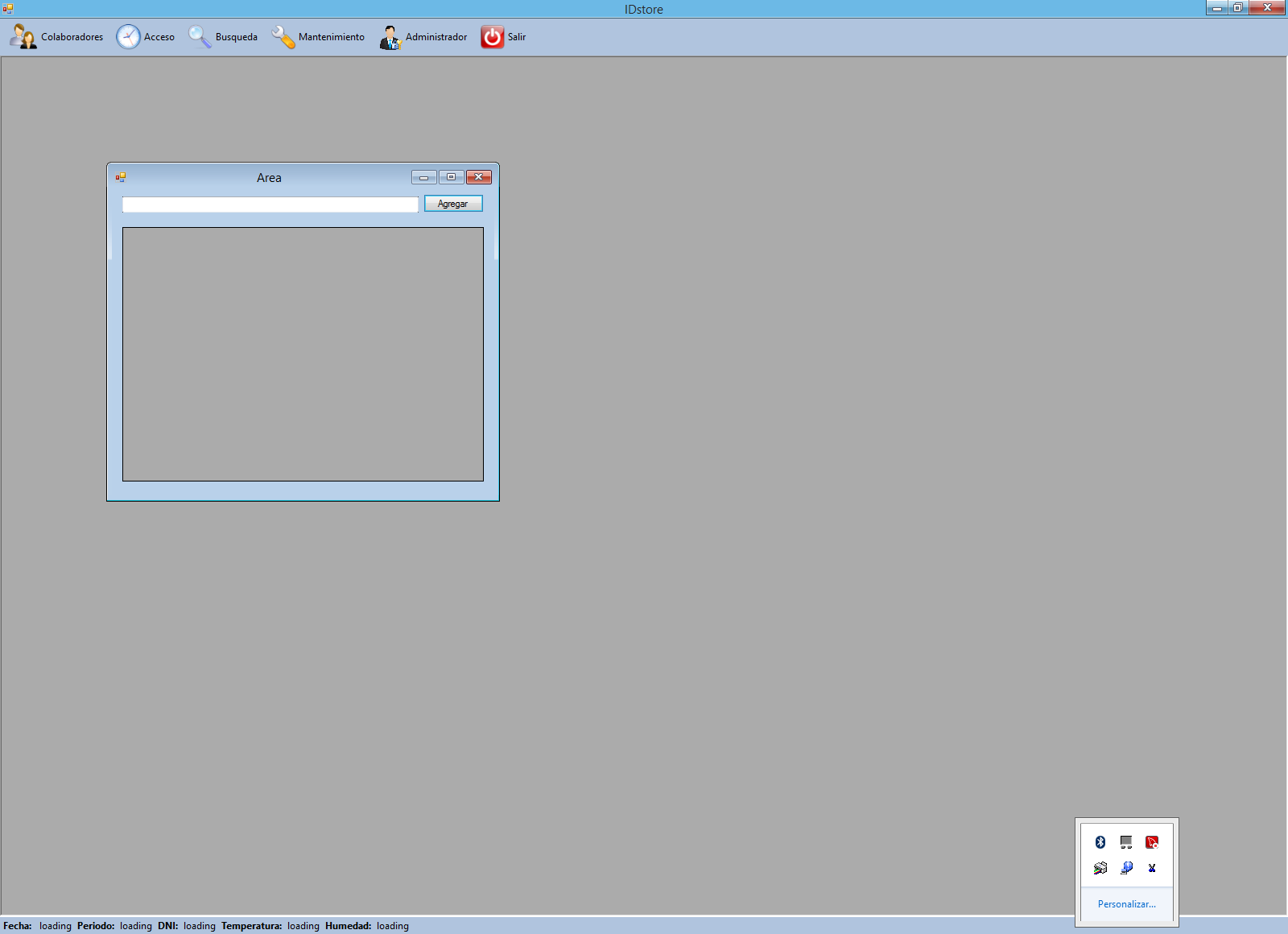


Figura 16 Ventana Para el Ingreso de Nuevas Areas

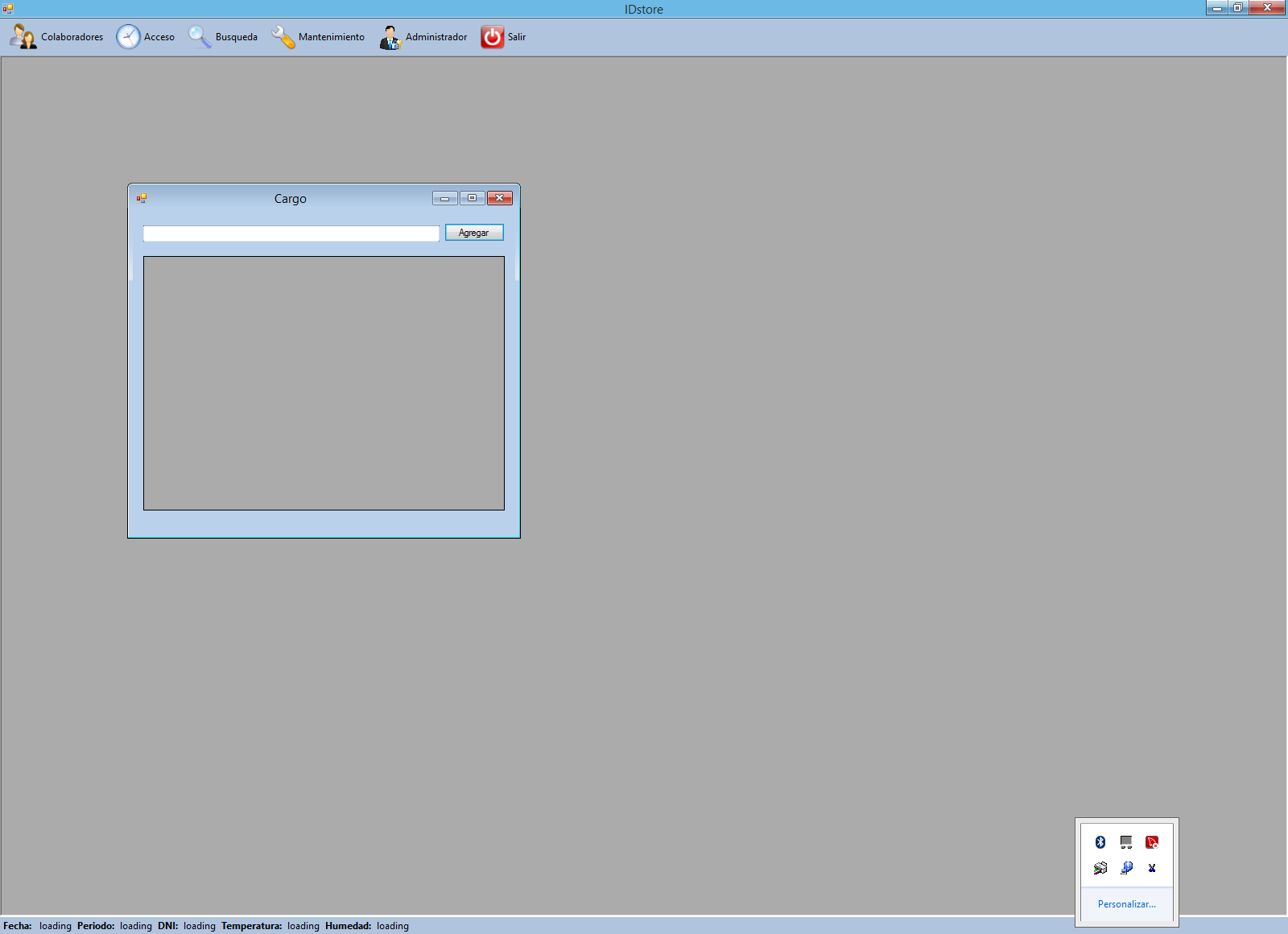


Figura 17 Ventana para el Ingreso de Nuevos Cargos

AxisCamara permite configurar la cámara Axis

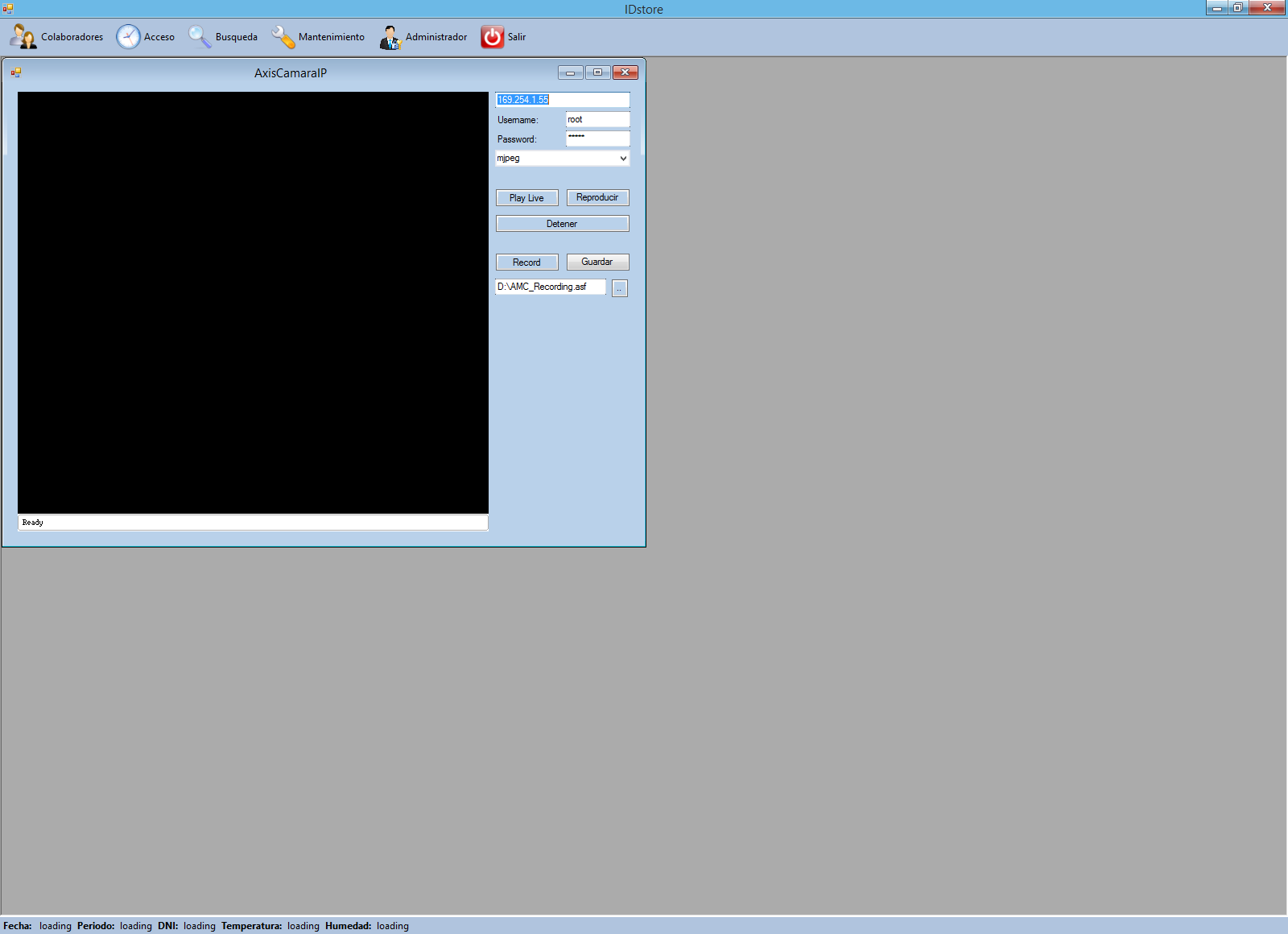


Figura 18 Ventana de Mantenimiento de Cámara Axis

Tanque solamente será utilizado cuando la empresa quiere controlar mas de un tanque de combustible.

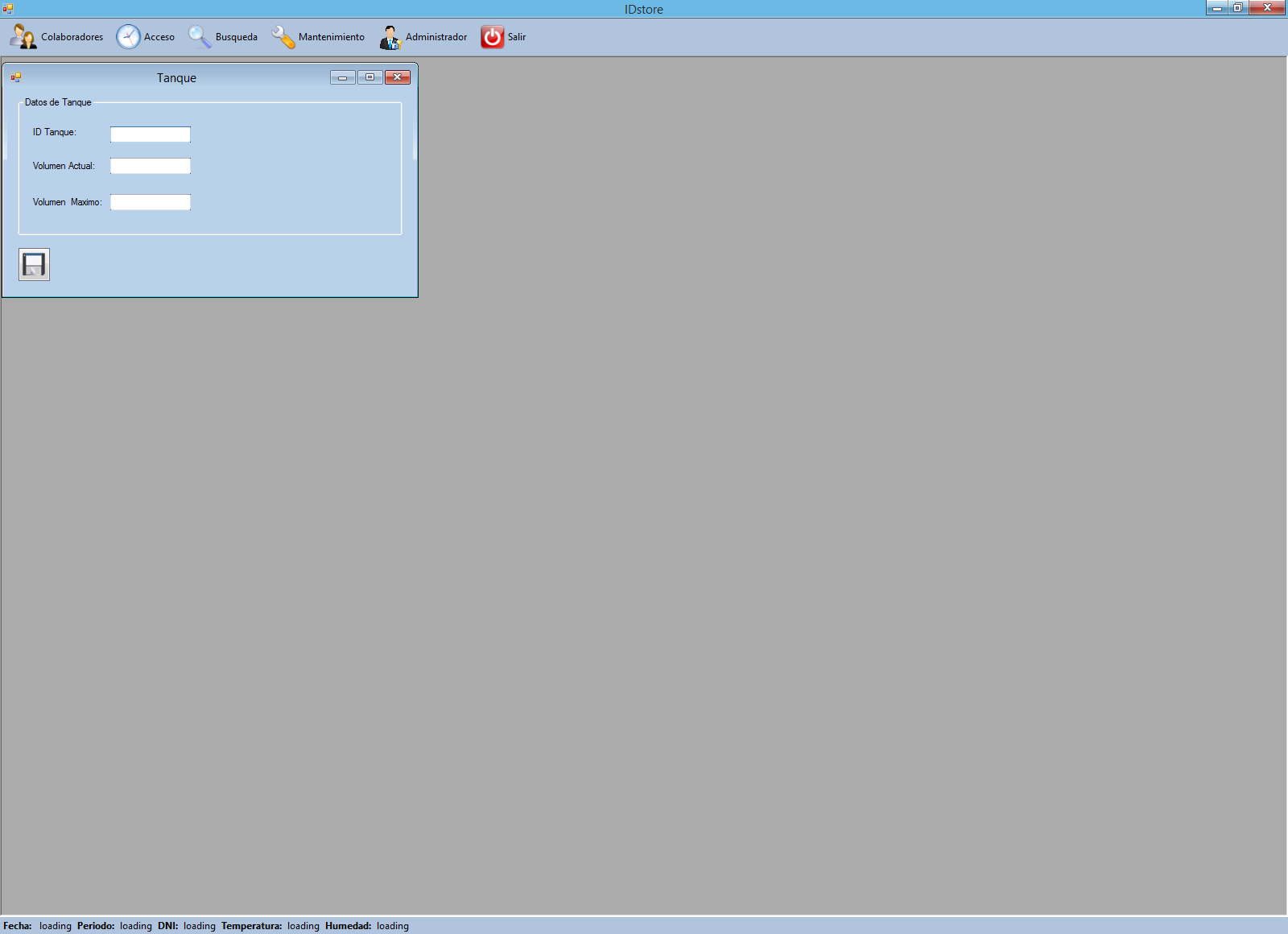


Figura 19 Ventana de Mantenimiento de Tanque

El formulario Sensor\_Temperatura\_y\_Humedad muestra la temperatura y humedad del depósito de combustible, además nos mostrara una alerta de acuerdo a la configuración realizada.

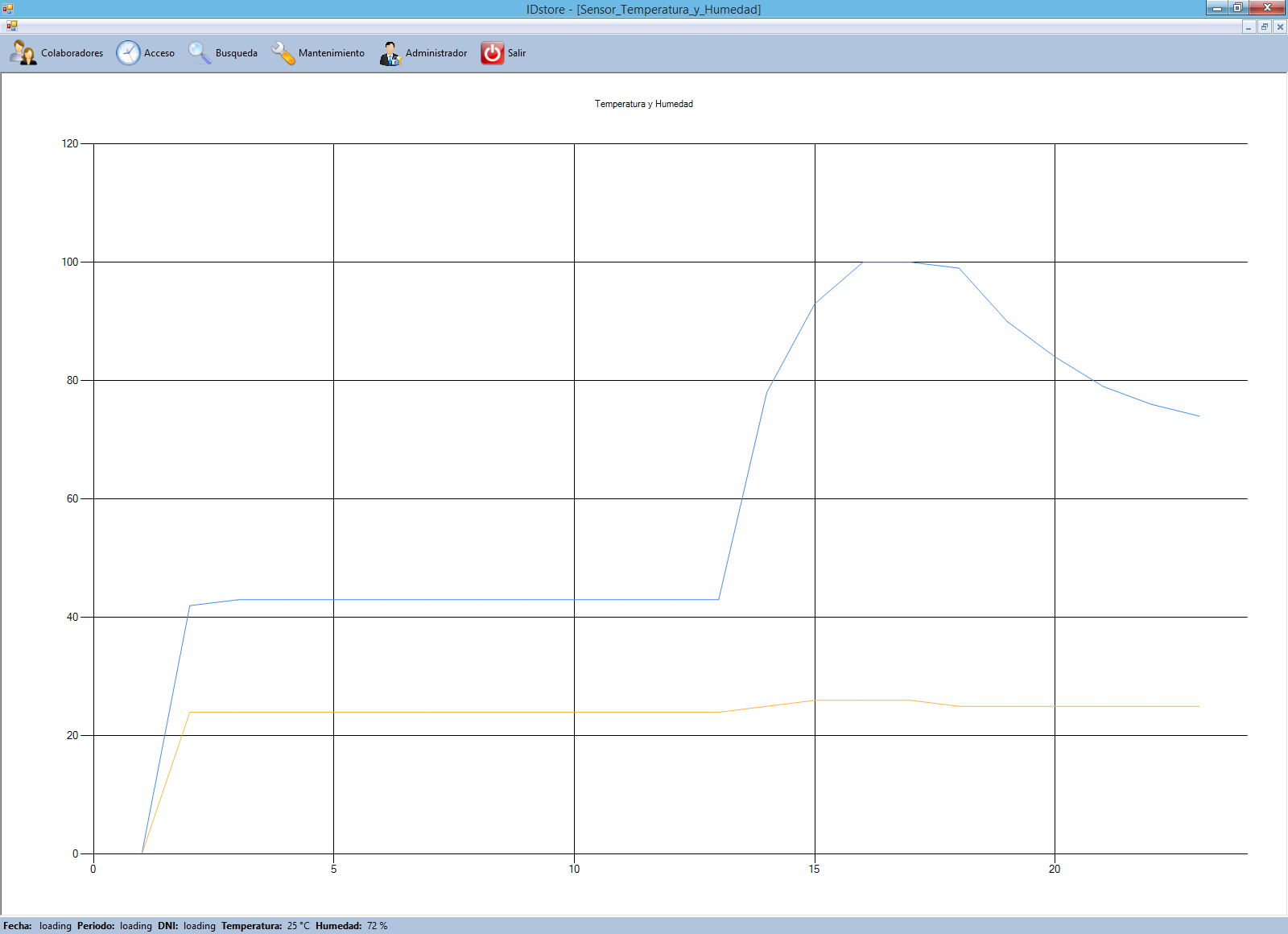


Figura 20 Ventana de Monitoreo de Sensor de Temperatura y Humedad

## 1.5.2 Hardware Empleado en la Implementación

Siguiendo la línea de cumplimiento de objetivos en el hardware, los componentes del hardware empleados en el presente trabajo son:

### 1.5.2.1 Lector de Código de Barras Symbol Ds 6608 2d

Los lectores de código de barras de permiten acelerar los procesos de lectura, alta y control de inventarios en clínicas médicas y hospitales con captura de datos superior y perfecta integración con otros sistemas de redes y de Sistemas de Información integrados con la lectura de códigos de barras.

Para cumplir los objetivos se dio por viable la utilización del hardware Lector de código de Barras (Motorola Symbol DS 6608) por las siguientes razones:

* Agilidad en la lectura de códigos 1D y 2D.
* Ante la pérdida de una Tarjeta de Identificación del personal el lector puede leer el código de barras del DNI, permitiendo su acceso rápidamente.
* La impresión de Tarjeta de Identificación es barato, ya que representaría un gasto de S/ 0.50 CU.
* Porcentaje de erro en la lectura es muy bajo, siendo este de 0.09 %.
* El lector de código de barras es fácil de conectar e instalar, ya que este cuenta con conexión USB.
* No necesita de una toma externa de corriente eléctrica, ya que su funcionamiento es similar el teclado de una computadora.
* El costo del hardware es viable S/ 450.00.



Figura 21 Lector de Código de Barras Symbol Ds 6608 2d.

### 1.5.2.2 Arduino Mega Rev 3 2560

Es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida.4 Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque que es ejecutado en la placa.

Las razones de su elección en el trabajo son:

* Compatible con los diversos lenguajes de programación (Phyton, Java, C#, Visual Basic, Mono, etc. ).
* El costo del hardware es económicamente viable S/ 150.00.
* El hardware ofrece la posibilidad de controlar múltiples dispositivos electrónicos: sensores de temperatura, motores, wifi, Ethernet, etc.

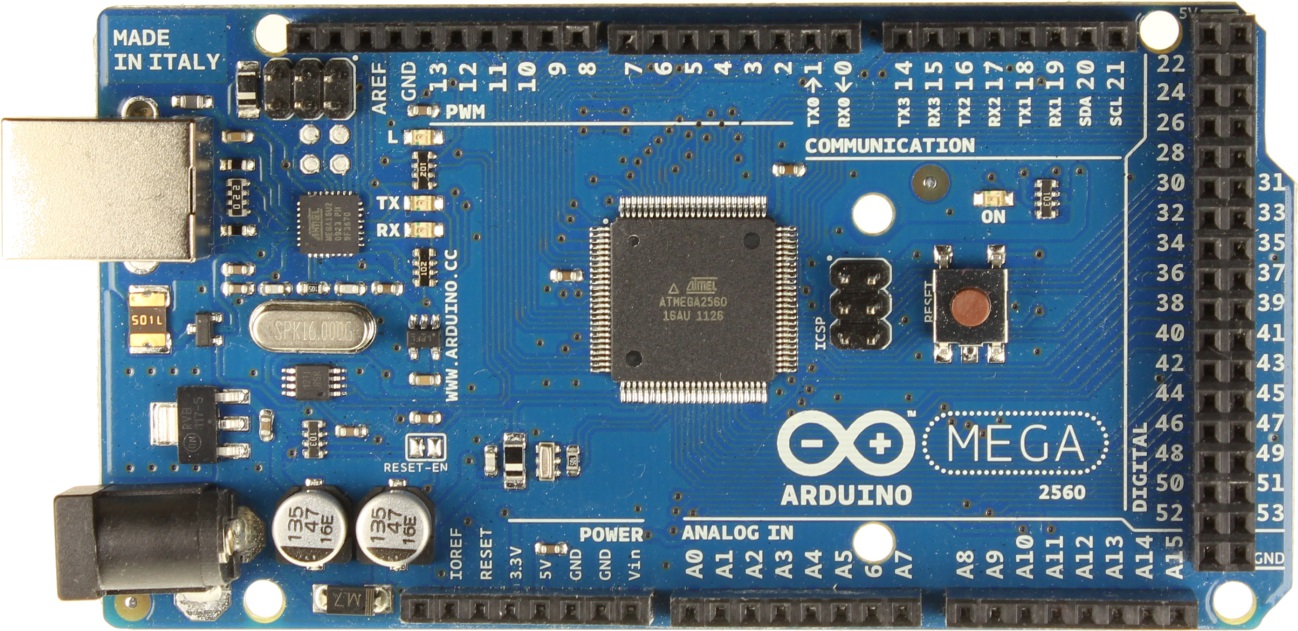


Figura 22 Arduino Mega Rev 3 2560.

### 1.5.2.3 Sensor de Temperatura y Humedad DHT 22

Sensor de humedad y temperatura de bajo costo. La salida suministrada es de tipo digital, no requiriéndose utilizar entradas analógicas. El sensor de humedad es de tipo capacitivo, mientras que la medición de temperatura se efectúa mediante un termistor.

Ideal PIC arduino, es un sensor de precisión de -40ºC a 80 ºC

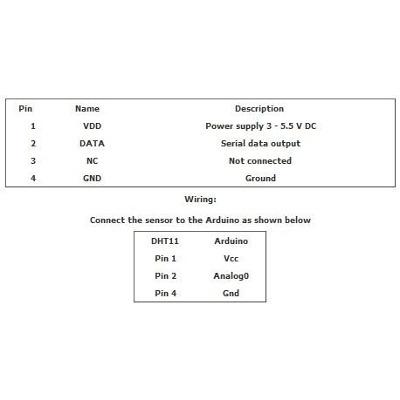
Longitud máxima del cable: 4mts

Muestreo: cada 2 segundos

VCC: 3 a 5,5v

**Detalles:**

**1. Tipo**: AM2302  
2. Precisión: 0.1  
3. Rango de Humedad: 0-100% de la humedad relativa  
4. rango de Temperatura: -40 ~ 80 °C  
5. Precisión de medición de Humedad: ± 2% de la humedad relativa  
6. Precisión de medida de la Temperature: ± 0,5 °C  
7. 4 Hilos o Pines (1 null - 3 funcionales)  
8. Potencia de muy bajo de consumo  
9. No incluye componentes adicionales  
10. Excelente estabilidad a largo plazo  
11. Todo Calibrado con salida digital  
12. Partes completamente intercambiables  
13. Buena transmisión de señales a distancia  
14. Medición de la humedad y temperatura relativa



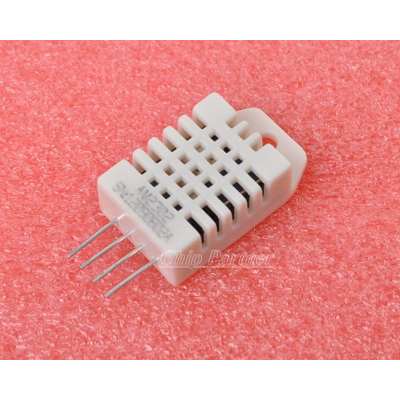
****

Figura 23 Sensor de Humedad y Temperatura DHT 22

### 1.5.2.4 Cámara IP AXIS M1054

La cámara de red AXIS M1054 cuenta con tecnología de barrido progresivo y proporciona vídeo con velocidad de imagen completa en resoluciones de hasta 1 megapíxel, incluida HDTV 720p. Puede ofrecer múltiples secuencias H.264 y Motion JPEG simultáneamente, optimizadas de modo individual para adaptarse a diversas necesidades de calidad y restricciones de ancho de banda.

La cámara AXIS M1054 presenta un sensor de infrarrojos pasivo (PIR) integrado para detectar el movimiento incluso en la oscuridad y un LED blanco para iluminar la escena si se produce un evento. Además admite puertos de E/S para conectar dispositivos externos.

La cámara AXIS M1054 está diseñada para una instalación sencilla y flexible y admite alimentación a través de Ethernet o mediante fuente de alimentación. La AXIS M1054 se puede instalar fácilmente en tiendas, restaurantes, hoteles y residencias particulares.

Caracteristicas

* Calidad HDTV
* Múltiples secuencias H.264
* Alimentación a través de Ethernet
* Sensor de infrarrojos pasivo (PIR) y LED de iluminación
* Micrófono y altavoz
* Puerto de E/S

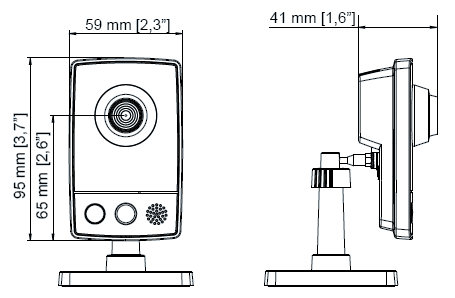


Figura 24 Esquema Fisicco Axis M1054



Figura 25 Cámara IP AXIS M1054

## 1.5.3 Diagrama de Montaje de Componentes Electrónicos

Se observa cómo deben quedar conectados los diferentes componentes electrónicos, para que el hardware interactúe correctamente con el software.

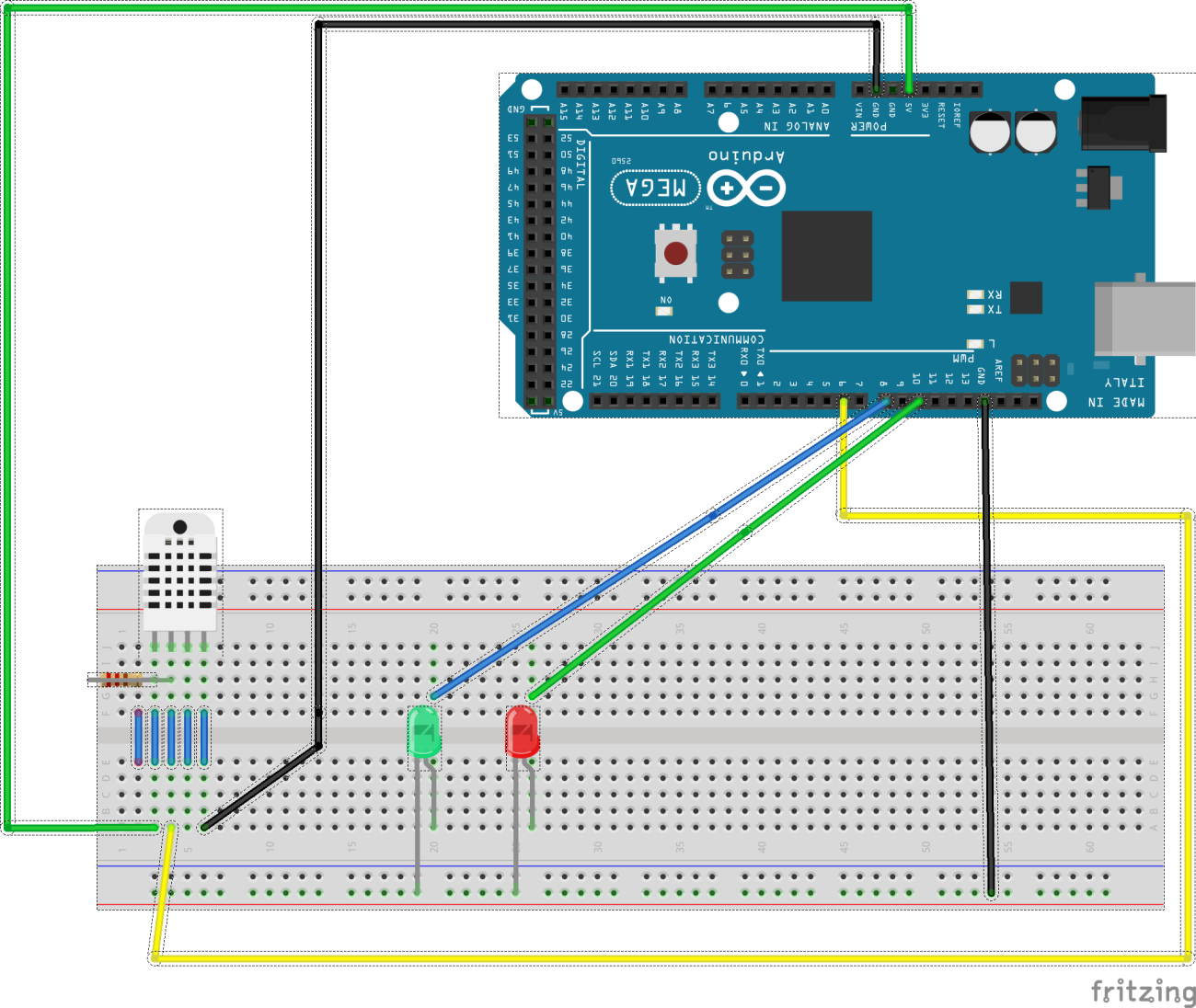


Figura 26 Diagrama de componentes electrónicos.



Figura 27 Foto de Conexiones Finales

## 1.5.4 Lista de Componentes Electrónicos

El diagrama que se muestra puede ser utilizado para seguir las conexiones que tienen los componentes electrónicos si solo se cuenta con el integrado at mega.

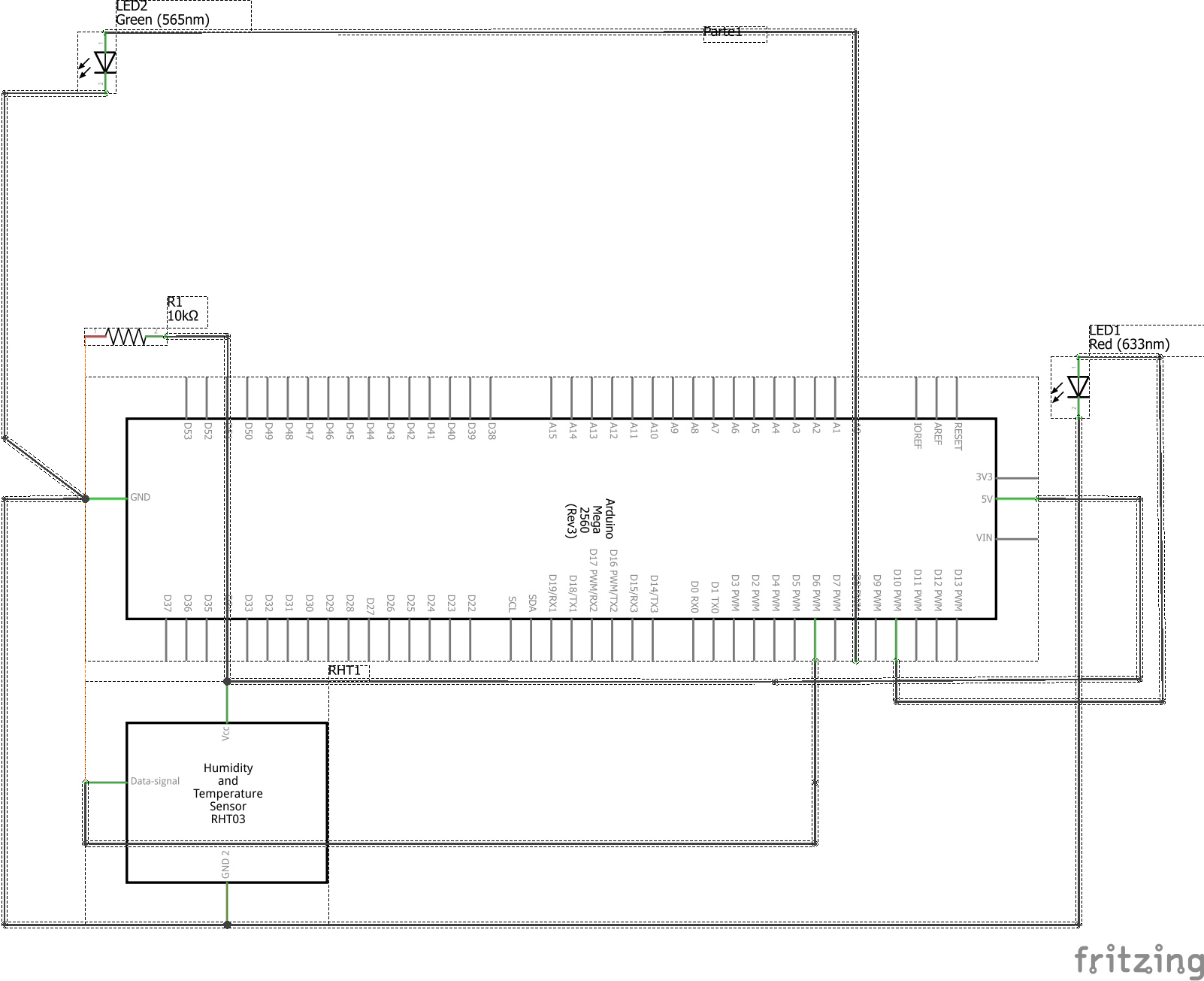


Figura 28 Vista de valores de componentes electrónicos.

## 1.5.5 Código Fuente del Hardware Arduino Mega 2560 Rev 3

Las líneas de código mas abajo permitir que el hardware capture mediante el sensor de humedad y temperatura las diferentes variaciones en humedad y temperatura, además nos permiten poder iluminar mediante diodos led el acceso permitido al colaborador.

#include <DHT22.h>

// Only used for sprintf

#include <stdio.h>

// Data wire is plugged into port 7 on the Arduino

// Connect a 4.7K resistor between VCC and the data pin (strong pullup)

#define DHT22\_PIN 6

// Setup a DHT22 instance

DHT22 myDHT22(DHT22\_PIN);

const int LED\_Rojo=8;

const int LED\_Verde=10;

int inByte = 0;

void **setup**(void)

{

  // start serial port

**Serial**.begin(9600);

   pinMode(LED\_Verde, OUTPUT);

    pinMode(LED\_Rojo, OUTPUT);

    digitalWrite(LED\_Verde, LOW);

    digitalWrite(LED\_Rojo, LOW);

}

void **loop**(void)

{

    if(**Serial**.available() > 0){//respuesta de puerto serie mayor a cero

        inByte = **Serial**.read(); //lectura del puerto serial

**Serial**.println(inByte);

        if(inByte == '0')

            digitalWrite(LED\_Verde, LOW);//apago led verde

        else if(inByte=='1')

           { digitalWrite(LED\_Verde, HIGH);// enciendo el led verde

            delay(500); // espera durante medio segundo

            digitalWrite(LED\_Verde, LOW);}// apago el led verde

        else if(inByte=='2')

            digitalWrite(LED\_Rojo, LOW);//apago el led rojo

        else if(inByte=='3')

           { digitalWrite(LED\_Rojo, HIGH);// enciendo el led rojo

             tone(6, 4978, 200);//emito un sonido en pwm 6, 200ms, Frecuencia 4978

            delay(500);// espera durante medio segundo

            digitalWrite(LED\_Rojo, LOW);

          }

    }

  DHT22\_ERROR\_t errorCode;

  delay(2000);

  errorCode = myDHT22.readData();

  switch(errorCode)

  {

    case DHT\_ERROR\_NONE:

     // Serial.print("Got Data ");

**Serial**.print(myDHT22.getHumidity());

**Serial**.print(",");

**Serial**.print(myDHT22.getTemperatureC());

**Serial**.println();

      break;

    case DHT\_ERROR\_CHECKSUM:

**Serial**.print("check sum error ");

**Serial**.print(myDHT22.getTemperatureC());

**Serial**.print("C ");

**Serial**.print(myDHT22.getHumidity());

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

      break;

    case DHT\_BUS\_HUNG:

**Serial**.println("BUS Hung ");

      break;

    case DHT\_ERROR\_NOT\_PRESENT:

**Serial**.println("Not Present ");

      break;

    case DHT\_ERROR\_ACK\_TOO\_LONG:

**Serial**.println("ACK time out ");

      break;

    case DHT\_ERROR\_SYNC\_TIMEOUT:

**Serial**.println("Sync Timeout ");

      break;

    case DHT\_ERROR\_DATA\_TIMEOUT:

**Serial**.println("Data Timeout ");

      break;

  }

}

# Conclusiones

Si tienes un proyecto de software, en el que debas interactuar con sistemas gestores de base de datos y con el mundo físico. La mejor opción es poder utilizar Oracle y Arduino como hardware libre.

Las ventajas que ofrece Oracle son conocidas a nivel mundial pero eso no significa descuidar nuestra Interfaz de usuario, ya que este también debe poder implementar un nivel de seguridad contra inyecciones sql, esto lo logramos con la programación en 4 capas.

# Sugerencias

Para empezar a trabaja con Oracle es necesario aprender el flujo de trabajo, de las diferentes instrucciones esto las dará una visión de la forma como funciona. Para mayor detalle visite la URL ([**http://www.oracleya.com.ar/**](http://www.oracleya.com.ar/)**).**

Si deseas programar hardware la mejor elección es arduino, ya que esta placa cuenta con una gran cantidad de sensores, actuadores, indicadores, GPS, WIFI, Ethenet, Giroscopios y una gran gama de dispositivos electrónicos. Arduino tiene una gran cantidad de librerías para los lenguajes de programación más conocidos como: C#, Java, Phython, C++, VB, JavaScript, etc. Para mayor información revisa la URL ([**http://arduino.cc/**](http://arduino.cc/)**).**

La seguridad electrónica cuenta con cámaras como la marca axis, usada regularmente por entidades financieras, si tu intención es poder explorar mas estos dispositivos visita la URL(<http://www.axis.com/>). Axis cuenta con una SDk para Visual Studio y nos permitirá programar aplicación para poder interactuar con estos dispositivos de seguridad.

Si deseas acceder al código fuente de la aplicación desarrollada, recuerda que lo puedes descargar desde el repositorio del proyecto en la URL (<https://github.com/Gsaico/IDSTORE>).

# Bibliografía

[**http://www.oracleya.com.ar/**](http://www.oracleya.com.ar/)

[**http://arduino.cc/**](http://arduino.cc/)

<http://www.axis.com/>

<http://sourcetreeapp.com/>

<http://windows.github.com/>