



ESCUELA SUPERIOR DE INGENIERÍA

Tecnología Avanzada de Bases de Datos

Ingeniería Informática

Javana

Autores:

Francisco Javier Molina Rojas

Adriana Maña Watson

Cádiz, 02 de 05 de 2023

TABD

1. Introducción	3
2. Requisitos funcionales y no funcionales.....	3
Contexto.....	3
Requisitos Funcionales	3
Requisitos no Funcionales	4
3. Diagrama de clases persistentes.....	4
4. Esquema lógico específica O/R	5
5. Diseño Físico	6
Tipo de datos.....	6
-- Socio.....	6
-- Adoptante	7
Secuencias.....	7
Tablas.....	8
--Tabla Animal.....	8
--Tabla Persona	8
--Tabla Voluntario	8
--Tabla Socio	8
--Tabla Adoptante	8
6. Disparadores y paquete.....	10
Paquete.....	10
Funciones	11
Procedimientos	14
Disparadores	23
7. Conclusiones.....	26
Anexo: Manual de Usuario	27
SQLDeveloper	27
Utilizar Visual Studio Code para Instalar PHP por XAMP (8.2)	29
Variables de entorno	30
OCI8	32
Referencias	35

1. Introducción

Este documento describe el trabajo realizado por Francisco Javier Molina Rojas y Adriana Maña Watson.

El proyecto está enfocado en realizar la base de datos de una protectora de animales, además de funciones para interactuar con ésta, y un front-end para los clientes y administradores.

2. Requisitos funcionales y no funcionales

Contexto

Con el aumento de abandonos, numerosos animales quedan expuestos a los peligros de la calle. Carreteras, abusadores... Un sinfín de riesgos.

Para la protección de animales nacen las protectoras y los refugios de animales.

Normalmente, estos trabajan con un conjunto diferente de usuarios y animales. Si los datos de estos no son tratados debidamente, puede haber muchas clases de problemas. Es bajo esta situación donde nace Javana.

Javana es una aplicación web que dispone de una base de datos Oracle que almacenará los datos respectivos a los usuarios, animales y servicios que pueden usar. El objetivo de Javana es facilitar el almacenamiento de estos datos, gracias a una interfaz amigable para cualquier tipo de protectora o refugio.

Requisitos Funcionales

La base de datos debe cumplir con los siguientes requisitos funcionales:

El administrador debe ser capaz de:

- Añadir a otros administradores
- Modificar contraseñas
- Eliminar a otros administradores

Una persona debe de ser capaz de:

- Ver los datos de los animales en adopción.
- Darse de alta como voluntario, acogiendo un animal.
- Darse de alta como socio.
- Adoptar un animal.

Un voluntario debe ser capaz de:

- Finalizar periodo de acogimiento con un animal.

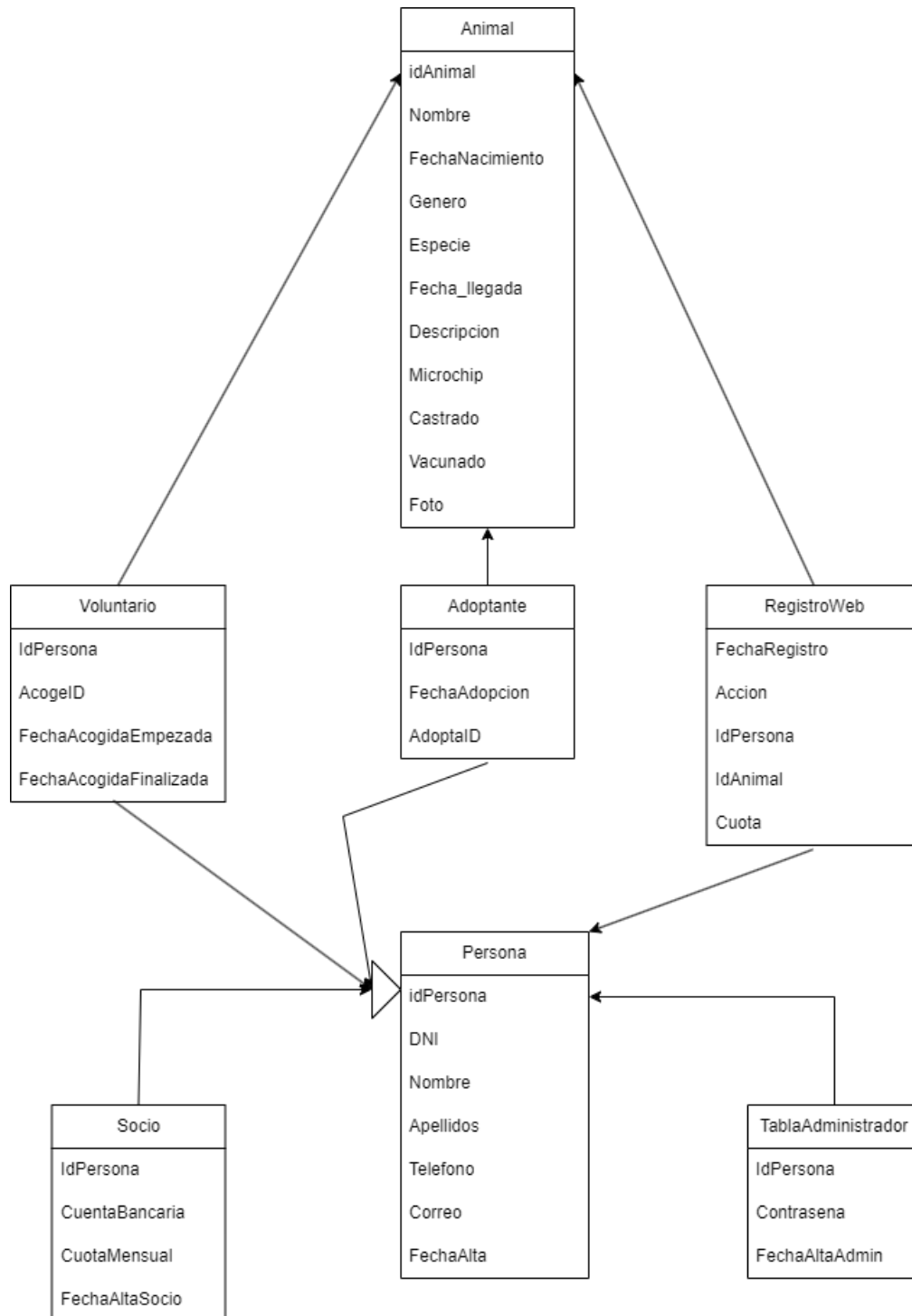
Un socio debe ser capaz de:

- Actualizar su cuota.
- Darse de Baja.

Requisitos no Funcionales

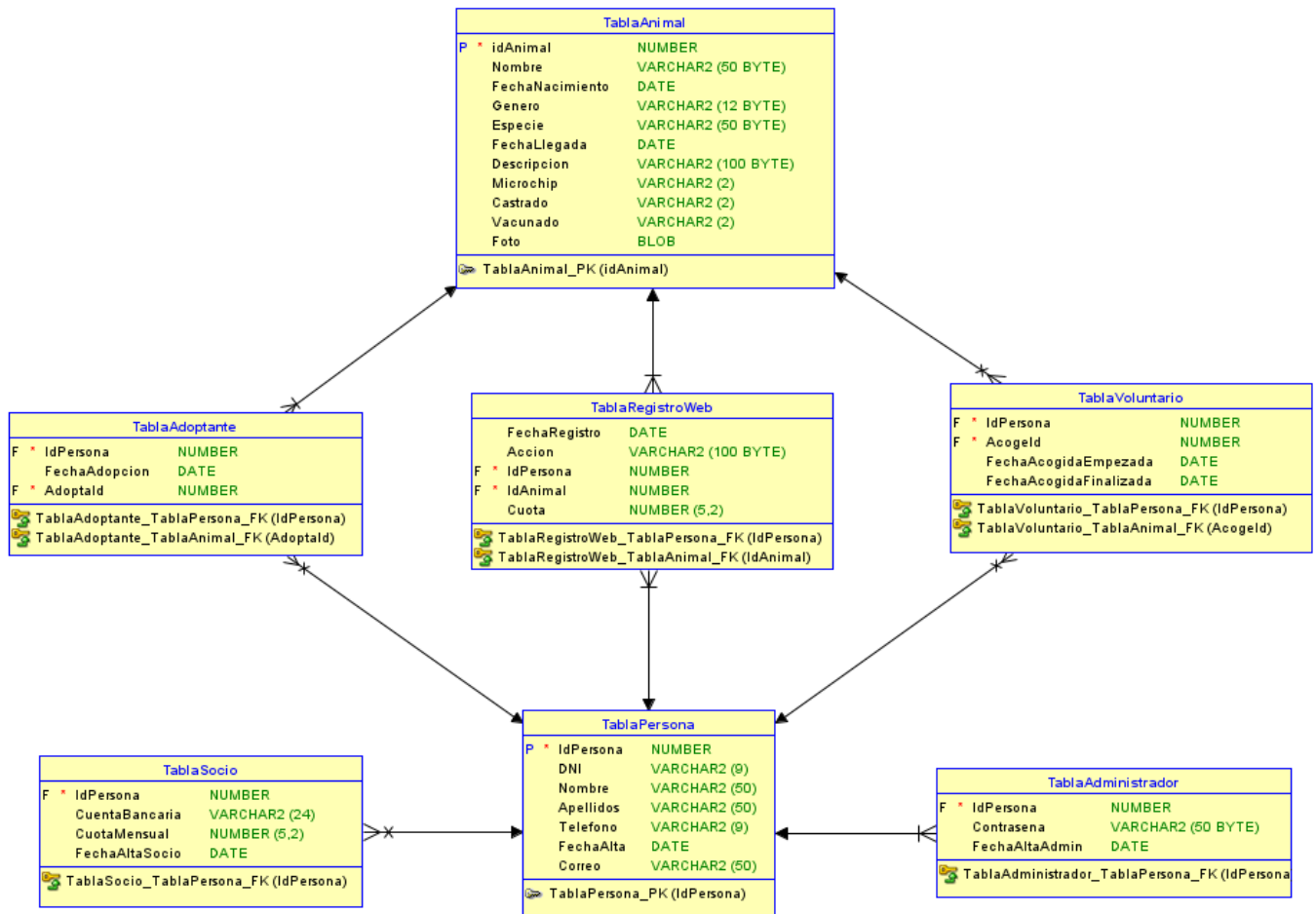
- Los animales pueden tener una foto asociada.
- Ofrecer a los socios distintas cuotas en forma de desplegable seleccionable.
- Tabla de registros globales de interacciones con el sistema.

3. Diagrama de clases persistentes



Aquí encontramos un diagrama del diseño lógico específico (UML) de la organización de la información necesaria y las relaciones entre las tablas que la conforman.

4. Esquema lógico específica O/R



Las tablas de la imagen anterior corresponden a la información que guarda el programa utilizado (sqldeveloper) sobre las tablas, sus restricciones y las relaciones entre ellas.

5. Diseño Físico

Tipo de datos

--Animal

```
CREATE OR REPLACE TYPE TipoAnimal AS OBJECT(  
    IdAnimal NUMBER,  
    Nombre VARCHAR2(50),  
    FechaNacimiento DATE,  
    Genero VARCHAR2(12),  
    Especie VARCHAR2(50),  
    FechaLlegada DATE,  
    Descripcion VARCHAR2(100),  
    Microchip VARCHAR2(2),  
    Castrado VARCHAR2(2),  
    Vacunado VARCHAR2(2),  
    Foto BLOB  
);  
/
```

-- Persona

```
CREATE OR REPLACE TYPE TipoPersona AS OBJECT(  
    IdPersona NUMBER,  
    DNI VARCHAR2(9),  
    Nombre VARCHAR2(50),  
    Apellidos VARCHAR2(50),  
    Telefono VARCHAR2(9),  
    FechaAlta DATE,  
    Correo VARCHAR2(50)  
);  
/
```

-- Voluntario

```
CREATE OR REPLACE TYPE TipoVoluntario AS OBJECT(  
    IdPersona NUMBER,  
    AcogeId NUMBER,  
    FechaAcogidaEmpezada DATE,  
    FechaAcogidaFinalizada DATE  
);  
/
```

-- Socio

```
CREATE OR REPLACE TYPE TipoSocio AS OBJECT(  
    IdPersona NUMBER,  
    CuentaBancaria VARCHAR2(24),  
    CuotaMensual NUMBER(5,2),  
    FechaAltaSocio DATE  
);  
/
```

```

-- Adoptante
CREATE OR REPLACE TYPE TipoAdoptante AS OBJECT(
    IdPersona NUMBER,
    FechaAdopcion DATE,
    AdoptaId NUMBER
);
/

-- Administrador
CREATE OR REPLACE TYPE TipoAdministrador AS OBJECT(
    IdPersona NUMBER,
    Contraseña VARCHAR2(50),
    FechaAltaAdmin DATE
);
/

-- Registro Web
CREATE OR REPLACE TYPE TipoRegistroWeb AS OBJECT(
    FechaRegistro DATE,
    Accion VARCHAR2(100),
    IdPersona NUMBER,
    IdAnimal NUMBER,
    Cuota NUMBER(5,2)
);
/

```

Secuencias

```

CREATE SEQUENCE secIdAnimal
INCREMENT BY 1
START WITH 1
MAXVALUE 50000
NOCACHE
NOCYCLE;
/

CREATE SEQUENCE secIdPersona
INCREMENT BY 1
START WITH 1
MAXVALUE 50000
NOCACHE
NOCYCLE;
/

```

Tablas

--Tabla Animal

```
CREATE TABLE TablaAnimal OF TipoAnimal
(
    CONSTRAINT PK_Animal PRIMARY KEY(IdAnimal),
    CONSTRAINT CE_NombreA CHECK (Nombre IS NOT NULL),
    CONSTRAINT CE_FechaNacimientoA CHECK (FechaNacimiento IS NOT NULL),
    CONSTRAINT CE_GeneroA CHECK (Genero IS NOT NULL),
    CONSTRAINT CE_EspecieA CHECK (Especie IS NOT NULL),
    CONSTRAINT CE_Fecha_llegadaA CHECK (FechaLlegada IS NOT NULL),
    CONSTRAINT CE_MicrochipA CHECK (Microchip IS NOT NULL),
    CONSTRAINT CE_CastradoA CHECK (Castrado IS NOT NULL),
    CONSTRAINT CE_VacunadoA CHECK (Vacunado IS NOT NULL)
);
/
```

--Tabla Persona

```
CREATE TABLE TablaPersona OF TipoPersona
(
    CONSTRAINT PK_Persona PRIMARY KEY(IdPersona),
    CONSTRAINT CE_DNIP CHECK (DNI IS NOT NULL),
    CONSTRAINT CE_NombreP CHECK (Nombre IS NOT NULL),
    CONSTRAINT CE_ApellidosP CHECK (Apellidos IS NOT NULL),
    CONSTRAINT CE_TelefonoP CHECK (Telefono IS NOT NULL),
    CONSTRAINT CE_CorreoP CHECK (Correo IS NOT NULL)
);
/
```

--Tabla Voluntario

```
CREATE TABLE TablaVoluntario OF TipoVoluntario
(
    CONSTRAINT FK_Voluntario FOREIGN KEY (IdPersona) REFERENCES TablaPersona(IdPersona)
);
/
```

--Tabla Socio

```
CREATE TABLE TablaSocio OF TipoSocio
(
    CONSTRAINT FK_Socio FOREIGN KEY (IdPersona) REFERENCES TablaPersona(IdPersona)
);
/
```

--Tabla Adoptante

```
CREATE TABLE TablaAdoptante OF TipoAdoptante
(
    CONSTRAINT FK_Adoptante FOREIGN KEY (IdPersona) REFERENCES TablaPersona(IdPersona)
);
/
```


-- Tabla Administrador

CREATE TABLE TablaAdministrador OF TipoAdministrador

```
(  
    CONSTRAINT FK_Administrador FOREIGN KEY (IdPersona) REFERENCES TablaPersona(IdPersona)  
);  
/
```

--Tabla de registros

CREATE TABLE TablaRegistroWeb OF TipoRegistroWeb

```
(  
    CONSTRAINT FK_RegistroWeb FOREIGN KEY (IdPersona) REFERENCES TablaPersona(IdPersona),  
    CONSTRAINT FK_RegistroWeb2 FOREIGN KEY (IdAnimal) REFERENCES TablaAnimal(IdAnimal)  
);  
/
```

6. Disparadores y paquete

Paquete

```
CREATE OR REPLACE PACKAGE PaqueteGlobal AS
    FUNCTION FAnimalAcogido(para_IdAnimal IN NUMBER) RETURN NUMBER;
    FUNCTION FAnimalAdoptado(para_IdAnimal IN NUMBER) RETURN NUMBER;
    FUNCTION FSumaCuotasMensuales RETURN NUMBER;
    FUNCTION EsAdmin(para_IdPersona IN NUMBER, para_contrasena IN VARCHAR2) RETURN NUMBER;
    FUNCTION EsAdmin2(para_IdPersona IN NUMBER) RETURN NUMBER;
    FUNCTION EsPersona(para_IdPersona IN NUMBER) RETURN NUMBER;
    FUNCTION EsSocio(para_IdPersona IN NUMBER) RETURN NUMBER;

    PROCEDURE AcogerAnimal(para_IdPersona IN NUMBER, para_IdAnimal IN NUMBER,
        FechaAcogidaEmpezada IN DATE);

    PROCEDURE AdoptarAnimal(para_IdPersona IN NUMBER, para_IdAnimal IN NUMBER, FechaAdopcion
        IN DATE);

    PROCEDURE FinalizarAcogidaAnimal(para_IdPersona IN NUMBER, para_IdAnimal IN NUMBER,
        para_FechaAcogidaFinalizada IN DATE);

    PROCEDURE DarDeAltaPersona(para_DNI IN VARCHAR2, para_Nombre IN VARCHAR2, para_Apellidos
        IN VARCHAR2, para_Telefono IN VARCHAR2, para_Correo IN VARCHAR2);

    PROCEDURE DarDeAltaAnimal(para_Nombre IN VARCHAR2, para_FechaNacimiento IN DATE,
        para_Genero IN VARCHAR2, para_Especie IN VARCHAR2, para_Fecha_llegada IN DATE,
        para_Descripcion IN VARCHAR2, para_Microchip IN VARCHAR2, para_Castrado IN VARCHAR2,
        para_Vacunado IN VARCHAR2, para_Foto IN BLOB);

    PROCEDURE CrearSocio(para_IdPersona IN NUMBER, para_CuentaBancaria IN VARCHAR2, para_Cuota
        IN NUMBER);

    PROCEDURE ActualizarCuotaSocio(para_IdPersona IN NUMBER, para_Cuota IN NUMBER);

    PROCEDURE DarDeBajaSocio(para_IdPersona IN NUMBER);

    PROCEDURE DarDeAltaAdministrador(para_IdPersona IN NUMBER, para_Contrasena IN VARCHAR2);

    PROCEDURE ActualizarAdministrador(para_IdPersona IN NUMBER, para_Contrasena IN VARCHAR2);

    PROCEDURE DarDeBajaAdministrador(para_IdPersona IN NUMBER);
END;
```

Funciones

FAnimalAcogido:

Devuelve 0 si el animal no se encuentra acogido, y 1 si se encuentra acogido.

```
FUNCTION FAnimalAcogido(para_IdAnimal IN NUMBER) RETURN NUMBER
IS
n NUMBER;
BEGIN
SELECT COUNT(*) into n FROM TablaVoluntario v WHERE v.AcogeId = para_IdAnimal AND
(v.FechaAcogidaFinalizada IS NULL OR v.FechaAcogidaFinalizada > SYSDATE);
RETURN n;
END;
```

FAnimalAdoptado:

Devuelve 0 si el animal no se encuentra adoptado, y 1 si se encuentra adoptado.

```
FUNCTION FAnimalAdoptado(para_IdAnimal IN NUMBER) RETURN NUMBER
IS
n NUMBER;
BEGIN
SELECT COUNT(*) INTO n FROM TablaAdoptante a WHERE a.AdoptaId = para_IdAnimal AND
a.FechaAdopcion IS NOT NULL;
RETURN n;
END;
```

FSumaCuotasMensuales:

Devuelve la suma total de las cuotas de todos los socios, siendo este el presupuesto del mes.

```
FUNCTION FSumaCuotasMensuales RETURN NUMBER
IS
nSumaCuotasMensuales NUMBER;
BEGIN
SELECT SUM(CuotaMensual) INTO nSumaCuotasMensuales FROM TablaSocio;
RETURN nSumaCuotasMensuales;
END;
```

EsAdmin:

Devuelve 0 si el id de la persona y su contraseña no coincide en la base de datos. Si coincide, devuelve cuantos.

```
FUNCTION EsAdmin(para_IdPersona IN NUMBER, para_contraseña IN VARCHAR2) RETURN NUMBER
IS
n NUMBER;
BEGIN
    SELECT COUNT(*) INTO n FROM TABLAADMINISTRADOR WHERE idPersona = para_IdPersona AND
        Contraseña = para_contraseña;
    RETURN n;
END;
```

EsAdmin2:

Devuelve 0 si el id de la persona no coincide en la tabla TablaAdministrador. Si coincide, devuelve cuantos.

```
FUNCTION EsAdmin(para_IdPersona IN NUMBER) RETURN NUMBER
IS
n NUMBER;
BEGIN
    SELECT COUNT(*) INTO n FROM TABLAADMINISTRADOR WHERE idPersona = para_IdPersona;
    RETURN n;
END;
```

EsPersona:

Devuelve 0 si el id de la persona no coincide en la tabla TablaPersona. Si coincide, devuelve cuantos coinciden.

```
FUNCTION EsPersona(para_IdPersona IN NUMBER) RETURN NUMBER
IS
n NUMBER;
BEGIN
    SELECT COUNT(*) INTO n FROM TablaPersona WHERE idPersona = para_IdPersona;
    RETURN n;
END;
```

EsSocio:

Devuelve 0 si el id del socio no coincide en la tabla TablaSocio. Si coincide, devuelve cuantos coinciden.

```
FUNCTION EsSocio(para_IdPersona IN NUMBER) RETURN NUMBER
IS
n NUMBER;
BEGIN
    SELECT COUNT(*) INTO n FROM TablaSocio WHERE idPersona = para_IdPersona;
    RETURN n;
END;
```

Procedimientos

AcogerAnimal:

Comprueba que el animal no esté ya acogido por otro voluntario ni que esté adoptado. Si estas condiciones se cumplen, se permitirá al voluntario acoger al animal.

```
PROCEDURE AcogerAnimal(para_IdPersona IN NUMBER, para_IdAnimal IN NUMBER,
FechaAcogidaEmpezada IN DATE) IS
n1 NUMBER;
n2 NUMBER;
n3 NUMBER;
AnimalYaAcogido EXCEPTION;
AnimalYaAdoptado EXCEPTION;
NoRegistrado EXCEPTION;
BEGIN
    n1:= FAnimalAcogido(para_IdAnimal);
    n2:= FAnimalAdoptado(para_IdAnimal);
    n3:= EsPersona(para_IdPersona);
    IF(n1 > 0) THEN
        RAISE AnimalYaAcogido;
    ELSIF(n2 > 0) THEN
        RAISE AnimalYaAdoptado;
    ELSIF(n3 = 0) THEN
        RAISE NoRegistrado;
    ELSE
        INSERT INTO TablaVoluntario (IdPersona, AcogeId, FechaAcogidaEmpezada,
FechaAcogidaFinalizada) VALUES (para_IdPersona, para_IdAnimal,
FechaAcogidaEmpezada, NULL);
    END IF;

EXCEPTION
    WHEN AnimalYaAcogido THEN
        DBMS_OUTPUT.PUT_LINE('El animal se encuentra acogido por otro voluntario');
        RAISE_APPLICATION_ERROR(-20500,'El animal se encuentra acogido por otro
voluntario');
    WHEN AnimalYaAdoptado THEN
        DBMS_OUTPUT.PUT_LINE('El animal ya ha sido adoptado');
        RAISE_APPLICATION_ERROR(-20501,'El animal ya ha sido adoptado');
    WHEN NoRegistrado THEN
        DBMS_OUTPUT.PUT_LINE('La persona no esta registrada');
        RAISE_APPLICATION_ERROR(-20509,'La persona no esta registrada');
END;
```

FinalizarAcogidaAnimal:

Comprueba que tuviera acogido al animal que quiere dejar de acoger en primer lugar. Si la fecha es superior a la fecha de adopción del animal, la fecha final de acogida será la fecha de la adopción. Si se cumple la condición, se registra la fecha en la que dejará de acoger al animal.

```
PROCEDURE FinalizarAcogidaAnimal(para_IdPersona IN NUMBER, para_IdAnimal IN NUMBER,
para_FechaAcogidaFinalizada IN DATE) IS
n1 NUMBER;
n2 NUMBER;
FechaAdop DATE;
AnimalNoAcogido EXCEPTION;
AnimalYaAdoptado EXCEPTION;
BEGIN
    SELECT COUNT(*) INTO n1 FROM TablaVoluntario v WHERE v.IdPersona = para_IdPersona
    AND v.AcogeId = para_IdAnimal AND v.FechaAcogidaFinalizada IS NULL;
    SELECT COUNT(*) INTO n2 FROM TablaAdoptante a WHERE a.AdoptaId = para_IdAnimal AND
    a.FechaAdopcion IS NOT NULL;
    IF(n2 > 0) THEN
        SELECT FechaAdopcion INTO FechaAdop FROM TablaAdoptante a WHERE
        a.AdoptaId = para_IdAnimal AND a.FechaAdopcion IS NOT NULL;
    IF (para_FechaAcogidaFinalizada < FechaAdop) THEN
        UPDATE TablaVoluntario SET FechaAcogidaFinalizada = para_FechaAcogidaFinalizada
        WHERE acogeId = para_idAnimal;
    ELSE
        UPDATE TablaVoluntario SET FechaAcogidaFinalizada = FechaAdop WHERE
        acogeId = para_idAnimal;
        RAISE AnimalYaAdoptado;
    END IF;
    ELSIF(n1 = 0) THEN
        RAISE AnimalNoAcogido;
    ELSE
        UPDATE TablaVoluntario SET FechaAcogidaFinalizada = para_FechaAcogidaFinalizada
        WHERE acogeId = para_idAnimal AND IdPersona = para_IdPersona AND
        FechaAcogidaFinalizada IS NULL;
    END IF;
EXCEPTION
    WHEN AnimalNoAcogido THEN
        DBMS_OUTPUT.PUT_LINE('El animal no se encuentra acogido');
        RAISE_APPLICATION_ERROR(-20503,'El animal no estaba acogido por la persona');
    WHEN AnimalYaAdoptado THEN
        DBMS_OUTPUT.PUT_LINE('El animal ya ha sido adoptado y ya no se encontrara
        acogido para ese entonces');
        RAISE_APPLICATION_ERROR(-20501,'El animal ya ha sido adoptado');
END;
```

AdoptarAnimal:

Comprueba que el animal no esté ya adoptado. Si se cumple esta condición, si el animal estaba anteriormente acogido por un voluntario, se actualizará la fecha de finalización de acogida en los datos del voluntario, y se permitirá al adoptante adoptar al animal.

```
PROCEDURE AdoptarAnimal(para_IdPersona IN NUMBER, para_IdAnimal IN NUMBER, FechaAdopcion
IN DATE) IS
n1 NUMBER;
n2 NUMBER;
n3 NUMBER;
AnimalYaAdoptado EXCEPTION;
NoRegistrado EXCEPTION;
BEGIN
    n1:= FAnimalAcogido(para_IdAnimal);
    n2:= FAnimalAdoptado(para_IdAnimal);
    n3:= EsPersona(para_IdPersona);
    IF(n1 > 0) THEN
        UPDATE TablaVoluntario SET FechaAcogidaFinalizada = FechaAdopcion WHERE acogeId =
        para_idAnimal AND FechaAcogidaFinalizada IS NULL;
        INSERT INTO TablaAdoptante (IdPersona,AdoptaId,FechaAdopcion) VALUES
        (para_IdPersona, para_IdAnimal, FechaAdopcion);
    ELSIF(n2 > 0) THEN
        RAISE AnimalYaAdoptado;
    ELSIF(n3 = 0) THEN
        RAISE NoRegistrado;
    ELSE
        INSERT INTO TablaAdoptante (IdPersona,AdoptaId,FechaAdopcion) VALUES
        (para_IdPersona, para_IdAnimal, FechaAdopcion);
    END IF;
EXCEPTION
    WHEN AnimalYaAdoptado THEN
        DBMS_OUTPUT.PUT_LINE('El animal ya ha sido adoptado');
        RAISE_APPLICATION_ERROR(-20501,'El animal ya ha sido adoptado');
    WHEN NoRegistrado THEN
        DBMS_OUTPUT.PUT_LINE('La persona no esta registrada');
        RAISE_APPLICATION_ERROR(-20509,'La persona no esta registrada');
END;
```


DarDeAltaAnimal:

Comprueba que el animal no esté ya registrado comparando que no exista la combinación de nombre y fecha de nacimiento iguales. Si se cumplen las condiciones, el animal quedará registrado en la base de datos.

```
PROCEDURE DarDeAltaAnimal(para_Nombre IN VARCHAR2, para_FechaNacimiento IN DATE,
para_Genero IN VARCHAR2, para_Especie IN VARCHAR2, para_Fecha_llegada IN DATE,
para_Descripcion IN VARCHAR2, para_Microchip IN VARCHAR2, para_Castrado IN VARCHAR2,
para_Vacunado IN VARCHAR2, para_Foto IN BLOB) IS
n1 NUMBER;
AnimalYaExiste EXCEPTION;
BEGIN
    SELECT COUNT(*) INTO n1 FROM TablaAnimal a WHERE a.Nombre = para_Nombre AND
    a.FechaNacimiento = para_FechaNacimiento;
    IF(n1 > 0) THEN
        RAISE AnimalYaExiste;
    ELSE
        INSERT INTO TablaAnimal (IdAnimal, Nombre, FechaNacimiento, Genero, Especie,
        FechaLlegada, Descripcion, Microchip, Castrado, Vacunado, Foto) VALUES
        (secIdAnimal.nextval, para_Nombre, para_FechaNacimiento, para_Genero, para_Especie,
        para_Fecha_llegada, para_Descripcion, para_Microchip, para_Castrado, para_Vacunado,
        para_Foto);
    END IF;
EXCEPTION
    WHEN AnimalYaExiste THEN
        DBMS_OUTPUT.PUT_LINE('El animal ya existe');
        RAISE_APPLICATION_ERROR(-20505, 'El animal ya existe');
END;
```

DarDeAltaPersona:

Comprueba que la persona no esté ya registrada comprobando mediante los DNI de la base de datos. Si se cumple la condición, se inserta a la persona en el sistema.

```
PROCEDURE DarDeAltaPersona(para_DNI IN VARCHAR2, para_Nombre IN VARCHAR2, para_Apellidos
IN VARCHAR2, para_Telefono IN VARCHAR2, para_Correo IN VARCHAR2) IS
n1 NUMBER;
PersonaYaExiste EXCEPTION;
BEGIN
    SELECT COUNT(*) INTO n1 FROM TablaPersona p WHERE p.DNI = para_DNI;
    IF(n1 > 0) THEN
        RAISE PersonaYaExiste;
    ELSE
        INSERT INTO TablaPersona (IdPersona,DNI,Nombre,Apellidos,Telefono,FechaAlta,Correo)
        VALUES (secIdPersona.nextval, para_DNI, para_Nombre, para_Apellidos, para_Telefono,
        SYSDATE, para_Correo);
    END IF;
EXCEPTION
    WHEN PersonaYaExiste THEN
        DBMS_OUTPUT.PUT_LINE('La persona ya existe');
        RAISE_APPLICATION_ERROR(-20504,'La persona ya existe');
END;
```

CrearSocio:

Comprueba que el id de la persona no esté ya registrada como socio. Si cumple la condición, se inserta al nuevo socio en el sistema.

```
PROCEDURE CrearSocio(para_IdPersona IN NUMBER,para_CuentaBancaria IN VARCHAR2, para_Cuota
IN NUMBER) IS
n1 NUMBER;
PersonaYaEsSocio EXCEPTION;
BEGIN
    n1 := EsSocio(para_IdPersona);
    IF(n1 > 0) THEN
        RAISE PersonaYaEsSocio;
    ELSE
        INSERT INTO TablaSocio (IdPersona,CuentaBancaria,CuotaMensual,FechaAltaSocio)
        VALUES (para_IdPersona,para_CuentaBancaria, para_Cuota, SYSDATE);
    END IF;
EXCEPTION
    WHEN PersonaYaEsSocio THEN
        DBMS_OUTPUT.PUT_LINE('La persona ya es socio');
        RAISE_APPLICATION_ERROR(-20506,'La persona ya es socio');
END;
```

ActualizarCuotaSocio:

Comprueba que la persona esté registrada como socio. Si se cumple la condición, permite al socio actualizar la cuota mensual.

```
PROCEDURE ActualizarCuotaSocio(para_IdPersona IN NUMBER, para_Cuota IN NUMBER) IS
n1 NUMBER;
PersonaNoEsSocio EXCEPTION;
BEGIN
    n1 := EsSocio(para_IdPersona);
    IF(n1 = 0) THEN
        RAISE PersonaNoEsSocio;
    ELSE
        UPDATE TablaSocio SET CuotaMensual = para_Cuota WHERE IdPersona = para_IdPersona;
    END IF;
EXCEPTION
    WHEN PersonaNoEsSocio THEN
        DBMS_OUTPUT.PUT_LINE('La persona no es socio');
        RAISE_APPLICATION_ERROR(-20507, 'La persona no es socio');
END;
```

DarDeBajaSocio:

Comprueba que el id de la persona no esté ya registrada como socio. Si cumple la condición, se da de baja al socio en la base de datos.

```
CREATE OR REPLACE PROCEDURE DarDeBajaSocio(para_IdPersona IN NUMBER) IS
n1 NUMBER;
PersonaNoEsSocio EXCEPTION;
BEGIN
    n1 := EsSocio(para_IdPersona);
    IF(n1 = 0) THEN
        RAISE PersonaNoEsSocio;
    ELSE
        DELETE FROM TablaSocio WHERE IdPersona = para_IdPersona;
    END IF;
EXCEPTION
    WHEN PersonaNoEsSocio THEN
        DBMS_OUTPUT.PUT_LINE('La persona no es socio');
        RAISE_APPLICATION_ERROR(-20507, 'La persona no es socio');
END;
```

DarDeAltaAdministrador:

Comprueba que el id de la persona esté registrada en la tabla TablaPersona y no esté ya registrada como Administrador. Si cumple la condición, se inserta al nuevo administrador en el sistema.

```
PROCEDURE DarDeAltaAdministrador(para_IdPersona IN NUMBER, para_Contrasena IN
VARCHAR2) IS
n1 NUMBER;
n2 NUMBER;
AdminYaExiste EXCEPTION;
NoRegistrado EXCEPTION;
BEGIN
  n1 := EsAdmin2(para_IdPersona);
  n2 := EsPersona(para_IdPersona);
  IF(n1 > 0) THEN
    RAISE AdminYaExiste;
  ELSIF(n2 = 0) THEN
    RAISE NoRegistrado;
  ELSE
    INSERT INTO TablaAdministrador (IdPersona,Contrasena,FechaAltaAdmin) VALUES
      (para_IdPersona, para_Contrasena, SYSDATE);
  END IF;
EXCEPTION
  WHEN AdminYaExiste THEN
    DBMS_OUTPUT.PUT_LINE('El admin ya existe');
    RAISE_APPLICATION_ERROR(-20508,'El admin ya existe');
  WHEN NoRegistrado THEN
    DBMS_OUTPUT.PUT_LINE('La persona no esta registrada');
    RAISE_APPLICATION_ERROR(-20509,'La persona no esta registrada');
END;
```

ActualizarAdministrador:

Comprueba que el id de la persona esté registrada en la tabla TablaPersona y esté ya registrada como Administrador. Si cumple la condición, se actualiza la contraseña del administrador recibido.

```
PROCEDURE ActualizarAdministrador(para_IdPersona IN NUMBER,para_Contrasena IN
VARCHAR2) IS
n1 NUMBER;
n2 NUMBER;
AdminNoExiste EXCEPTION;
NoRegistrado EXCEPTION;
BEGIN
    n1 := EsAdmin2(para_IdPersona);
    n2 := EsPersona(para_IdPersona);
    IF(n1 = 0) THEN
        RAISE AdminNoExiste;
    ELSIF(n2 = 0) THEN
        RAISE NoRegistrado;
    ELSE
        UPDATE TABLAADMINISTRADOR SET Contraseña = para_Contrasena WHERE
        IdPersona = para_IdPersona;
    END IF;
EXCEPTION
    WHEN AdminNoExiste THEN
        DBMS_OUTPUT.PUT_LINE('El admin No existe');
        RAISE_APPLICATION_ERROR(-20510,'El admin No existe');
    WHEN NoRegistrado THEN
        DBMS_OUTPUT.PUT_LINE('La persona no esta registrada');
        RAISE_APPLICATION_ERROR(-20509,'La persona no esta registrada');
END;
```

DarDeBajaAdministrador:

Comprueba que el id de la persona esté registrada en la tabla TablaPersona y esté ya registrada como Administrador. Si cumple la condición, se da de baja al administrador recibido.

```
PROCEDURE DarDeBajaAdministrador(para_IdPersona IN NUMBER) IS
n1 NUMBER;
n2 NUMBER;
AdminNoExiste EXCEPTION;
NoRegistrado EXCEPTION;
BEGIN
    n1 := EsAdmin2(para_IdPersona);
    n2 := EsPersona(para_IdPersona);
    IF(n1 = 0) THEN
        RAISE AdminNoExiste;
    ELSIF(n2 = 0) THEN
        RAISE NoRegistrado;
    ELSE
        DELETE FROM TABLAADMINISTRADOR WHERE IdPersona = para_IdPersona;
    END IF;
EXCEPTION
    WHEN AdminNoExiste THEN
        DBMS_OUTPUT.PUT_LINE('El admin No existe');
        RAISE_APPLICATION_ERROR(-20510,'El admin No existe');
    WHEN NoRegistrado THEN
        DBMS_OUTPUT.PUT_LINE('La persona no esta registrada');
        RAISE_APPLICATION_ERROR(-20509,'La persona no esta registrada');
END;
```

Disparadores

RegistroPersonasT:

Trigger para insertar en la tabla TablaRegistroWeb cuando se inserte un registro de la tabla TablaPersona.

```
CREATE OR REPLACE TRIGGER RegistroPersonasT --Creamos trigger
AFTER INSERT OR UPDATE OR DELETE ON TablaPersona FOR EACH ROW
    --Despues de que se inserte un registro de la tabla TablaPersona
DECLARE
BEGIN
    IF INSERTING THEN --Cuando se inserte una fila:
        INSERT INTO TablaRegistroWeb (FechaRegistro, Accion, IdPersona, IdAnimal, Cuota)
        VALUES (SYSDATE, 'Se ha registrado una nueva persona en el sistema',
        :NEW.IdPersona, NULL, NULL);
        --Insertamos en la tabla TablaRegistroWeb los datos de la persona insertada
    END IF;
EXCEPTION -- Apartado de excepciones
    -- Al tratarse de un trigger con solo operaciones insert, no existen muchas
    excepciones que se puedan dar.
    -- por lo que decidimos que si se produce una excepcion no esperada, se muestre un
    mensaje de error.
    WHEN OTHERS THEN -- Cuando ocurra una excepcion no esperada
        DBMS_OUTPUT.PUT_LINE('Ha ocurrido un error, revisa tu operacion y comprueba que los
        datos son correctos'); --Mostramos un mensaje de error
END;
/
```

RegistroSocioT:

Trigger para insertar en la tabla TablaRegistroWeb cuando se inserte, actualice o borre un registro de la tabla TablaSocio.

```
CREATE OR REPLACE TRIGGER RegistroSocioT
AFTER INSERT OR UPDATE OR DELETE ON TablaSocio FOR EACH ROW
DECLARE
BEGIN
    IF INSERTING THEN
        INSERT INTO TablaRegistroWeb (FechaRegistro, Accion, IdPersona, IdAnimal,
        Cuota) VALUES (SYSDATE, 'Se ha registrado un nuevo socio en el sistema',
        :NEW.IdPersona, NULL, :NEW.CuotaMensual);
    ELSIF UPDATING THEN
        INSERT INTO TablaRegistroWeb (FechaRegistro, Accion, IdPersona, IdAnimal,
        Cuota) VALUES (SYSDATE, 'Se ha actualizado un socio en el sistema',
        :NEW.IdPersona, NULL, :NEW.CuotaMensual);
    ELSIF DELETING THEN
        INSERT INTO TablaRegistroWeb (FechaRegistro, Accion, IdPersona, IdAnimal,
        Cuota) VALUES (SYSDATE, 'Se ha borrado un socio en el sistema', :OLD.IdPersona,
        NULL, NULL);
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Ha ocurrido un error, revisa tu operacion y comprueba que
        los datos son correctos');
END;
/
```

RegistroAdoptantesT:

Trigger para insertar en la tabla TablaRegistroWeb cuando se inserte un registro de la TablaAdoptante.

```
CREATE OR REPLACE TRIGGER RegistroAdoptantesT
AFTER INSERT ON TablaAdoptante FOR EACH ROW
DECLARE
BEGIN
    INSERT INTO TablaRegistroWeb (FechaRegistro, Accion, IdPersona, IdAnimal, Cuota) VALUES
    (SYSDATE, 'Se ha registrado un nuevo adoptante en el sistema', :NEW.IdPersona,
    :NEW.AdoptaId, NULL);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Ha ocurrido un error, revisa tu operacion y comprueba que los
        datos son correctos');
END;
/
```


RegistroVoluntariosT:

Trigger para insertar en la tabla TablaRegistroWeb cuando se inserte un registro de la TablaVoluntario.

```
CREATE OR REPLACE TRIGGER RegistroVoluntariosT
AFTER INSERT OR UPDATE OR DELETE ON TablaVoluntario FOR EACH ROW
DECLARE
BEGIN
    IF INSERTING THEN
        INSERT INTO TablaRegistroWeb (FechaRegistro, Accion, IdPersona, IdAnimal, Cuota)
VALUES (SYSDATE, 'Se ha registrado un nuevo voluntario en el sistema', :NEW.IdPersona,
:NEW.AcogeId, NULL);
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Ha ocurrido un error, revisa tu operacion y comprueba que los
datos son correctos');
END;
/
```

RegistroAdministradoresT:

Trigger para insertar en la tabla TablaRegistroWeb cuando se inserte, actualice o borre un registro de la tabla TablaAdministrador.

```
CREATE OR REPLACE TRIGGER RegistroAdministradoresT
AFTER INSERT OR UPDATE OR DELETE ON TablaAdministrador FOR EACH ROW
BEGIN
    IF INSERTING THEN
        INSERT INTO TablaRegistroWeb (FechaRegistro, Accion, IdPersona, IdAnimal, Cuota)
VALUES (SYSDATE, 'Se ha registrado un nuevo administrador en el sistema',
:NEW.IdPersona, NULL, NULL);

    ELSIF UPDATING THEN
        INSERT INTO TablaRegistroWeb (FechaRegistro, Accion, IdPersona, IdAnimal, Cuota)
VALUES (SYSDATE, 'Se ha actualizado un administrador en el sistema',
:NEW.IdPersona, NULL, NULL);

    ELSIF DELETING THEN
        INSERT INTO TablaRegistroWeb (FechaRegistro, Accion, IdPersona, IdAnimal, Cuota)
VALUES (SYSDATE, 'Se ha borrado un administrador en el sistema', :OLD.IdPersona,
NULL, NULL);
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Ha ocurrido un error, revisa tu operacion y comprueba que los
datos son correctos');
END;
/
```

7. Conclusiones

Este trabajo nos ha servido para darnos cuenta de que tan importante son las bases de datos en la actualidad. Sin estas, muchos de los servicios que hoy damos por hecho, no podrían existir.

Gracias a las bases de datos se nos permite:

- Gestionar la información de una manera eficiente y consistente.
- Controlar la redundancia de datos.
- Compartir la información de una manera rápida.
- Controlar fallos que pueden existir en esta.
- Manejar grandes cantidades de información.
- Automatizar procesos.
- Muchas otras acciones...

Usando bases de datos O/R, conseguimos tener un extra de eficiencia y control de los datos al tener la ventaja del uso de objetos.

El hecho de no tener todo el conocimiento/tiempo necesario para poder realizar todo lo que hubiéramos querido, afectó a la motivación del grupo, pues después de realizar las funciones, procedimientos y construcción de la base de datos, no sabíamos como realizar ninguna de estas en PHP.

Sin embargo, después de mucho fallo y error, logramos construir una aplicación funcional y usable.

En conclusión, realizar este proyecto nos ha servido para mejorar nuestro conocimiento de bases de datos, trabajar bajo presión y gestionar los tiempos.

Anexo: Manual de Usuario

SQLDeveloper

- Para poder ejecutar los archivos SQL, debemos tener instalado SQLdeveloper.
- Todos los archivos de este proyecto deberán encontrarse en la misma carpeta.
- Para que el proyecto funcione, es necesario crear un usuario con los nombre específicos.

Instalación y creación de conexiones

- Conectarse a Oracle desde la cuenta SYSTEM. Por ejemplo, desde el símbolo del sistema: sqlplus system y escribir la contraseña cuando se solicite.
- Ejecute lo siguiente para moverse a la base de datos XEPDB1

```
ALTER SESSION SET CONTAINER = XEPDB1;
```

Aparecerá el mensaje, "sesión modificada".

- Crear un usuario con nombre de usuario "javivimol" y contraseña "admin".

```
CREATE USER javivimol IDENTIFIED BY admin
DEFAULT TABLESPACE USERS
QUOTA UNLIMITED ON USERS
TEMPORARY TABLESPACE TEMP
CONTAINER = CURRENT;
```

Aparecerá el mensaje "usuario creado".

- Ejecute lo siguiente para conceder los permisos necesarios para crear sesión, crear objetos y crear software en la base de datos al usuario recién creado.

```
GRANT CONNECT, RESOURCE TO javivimol CONTAINER = CURRENT;
```

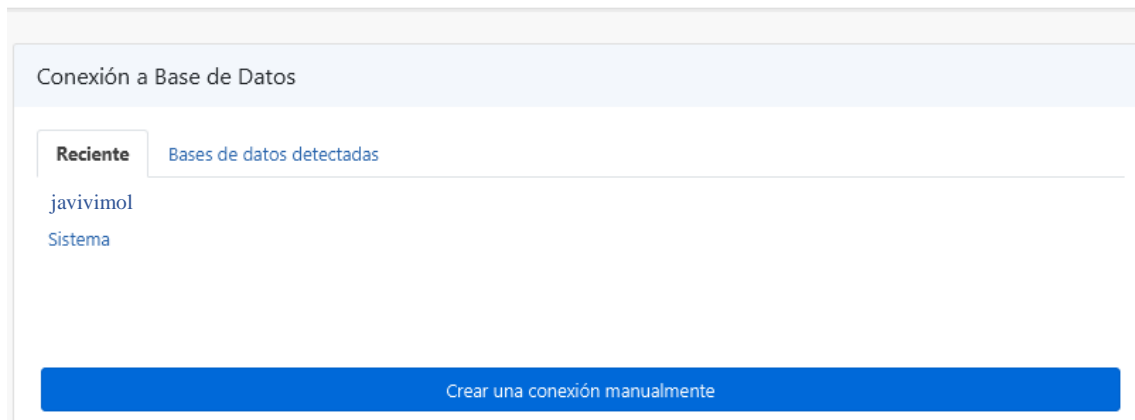
Aparecerá el mensaje "concesión terminada correctamente".

- Descargar el producto **Oracle SQL Developer** de <https://www.oracle.com/tools/downloads/sqldev-downloads.html>.

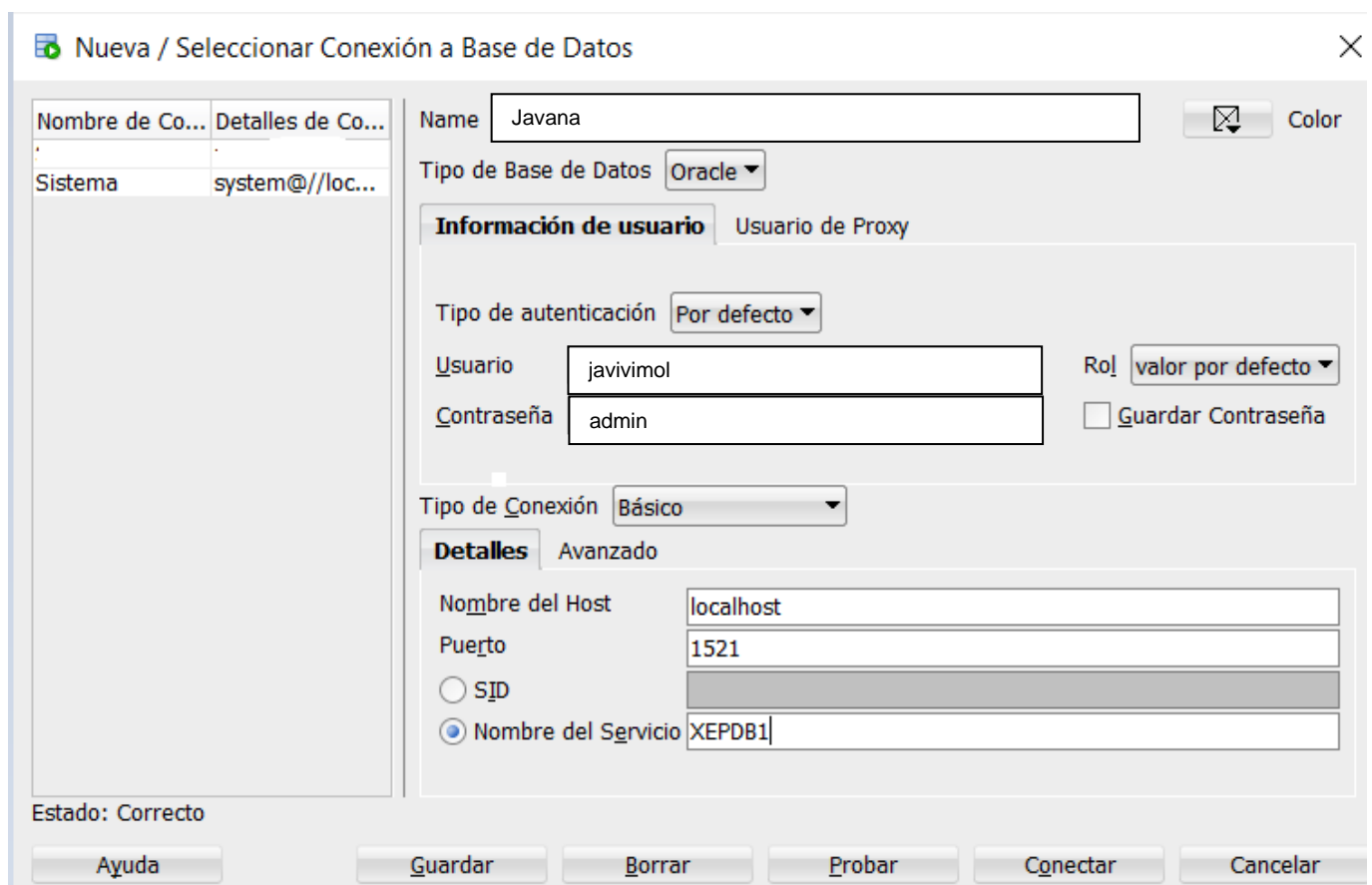
Para la descarga, es necesario tener cuenta de Oracle.

Una vez descargado, descomprimir y ejecutar el programa.

- Ahora hay que crear las *conexiones para el sistema y para el usuario*. En la página de bienvenida, hacemos clic en "**Crear una conexión manualmente**". (Damos por hecho que ya está creada la conexión al sistema).



- Ponemos los datos de la imagen, y le damos a "*Guardar*". Aparecerá la nueva conexión creada para el usuario.
 - En *name* escribimos el nombre que le queremos dar a la conexión.
 - El *nombre de usuario* y la *clave* siguientes: **jativimol** y **admin**. Esto es necesario para que funcione la conexión php correctamente.
 - En "**nombre del servicio**" pondremos la base de datos conectable **XEPDB1**.



Ahora nos aparecerán, en la pantalla de bienvenida, las conexiones creadas anteriormente.

Comenzar el proyecto:

1. Entramos al usuario creado.
2. Pulsamos “archivo”, y pulsamos “abrir”
3. Insertamos el archivo “llamadaInicial.sql”
4. Lo ejecutamos.

Utilizar Visual Studio Code para Instalar PHP por XAMP (8.2)

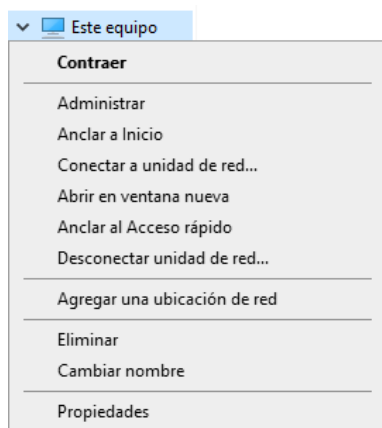
1. Descargar: <https://www.apachefriends.org/es/index.html> para S.O. correspondiente.
2. Entrar al archivo xamp en C: (Disco del sistema operativo)
3. Darle a xampp-control.exe
4. Pulsar START junto a Apache
5. Abrir Visual Studio Code
6. Buscar en extensions: “PHP extensión Pack” de Xdebug, “PHP IntelliSense” de Damjan Cvtko y “PHP Server” de brapifra. Se recomienda también “PHP Debug” de Xdebug.
7. Tras la instalación, cerramos y abrimos Visual Studio Code.
8. Abrimos File > Preferences > Settings.
9. Pulsamos Open Settings



10. Añadir la siguiente línea si no se encuentra:
"php.validate.executablePath": "c:\\php\\php.exe" (EN WINDOWS)
Si no es Windows, entramos a <https://code.visualstudio.com/docs/languages/php> y buscamos el correspondiente.

Variables de entorno

11. Entramos a C:\xampp\php y copiamos la ruta.
12. En el explorador de archivos, le damos a propiedades sobre Este Equipo.



13. Buscamos configuración avanzada del sistema.

Opciones de configuración
relacionadas

Configuración de BitLocker

Administrador de dispositivos

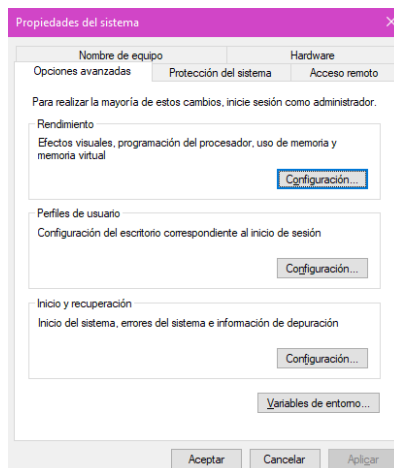
Escritorio remoto

Protección del sistema

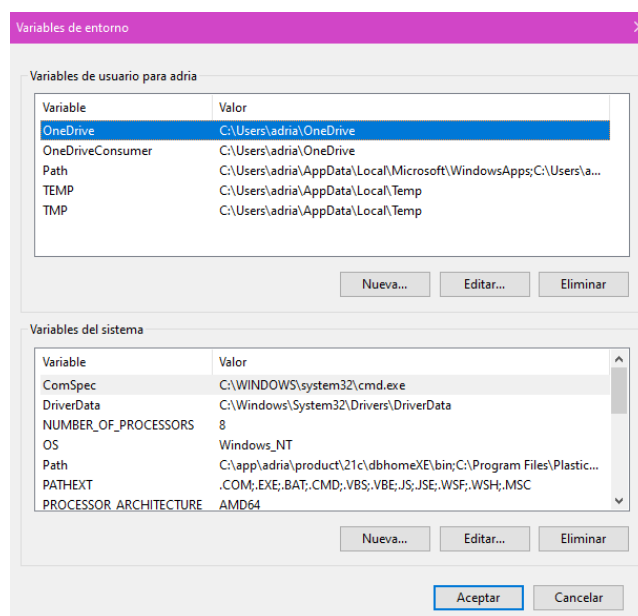
Configuración avanzada del sistema

Cambiar el nombre de este equipo
(avanzado)

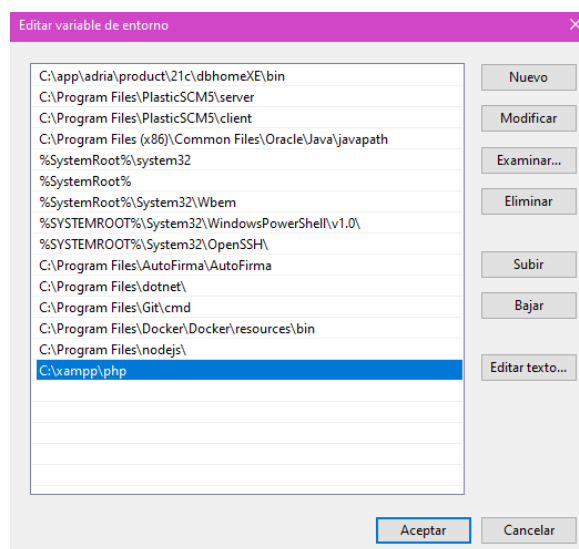
14. Le damos a variables de entorno.



15. Buscamos Path y le damos doble clic.



16. Añadimos xampp dándole a Nuevo y añadiendo la ruta.



17. Guardamos (Aceptar).

18. Reiniciamos el sistema.

19. Descargar las carpetas del proyecto si no estaban descargadas ya.

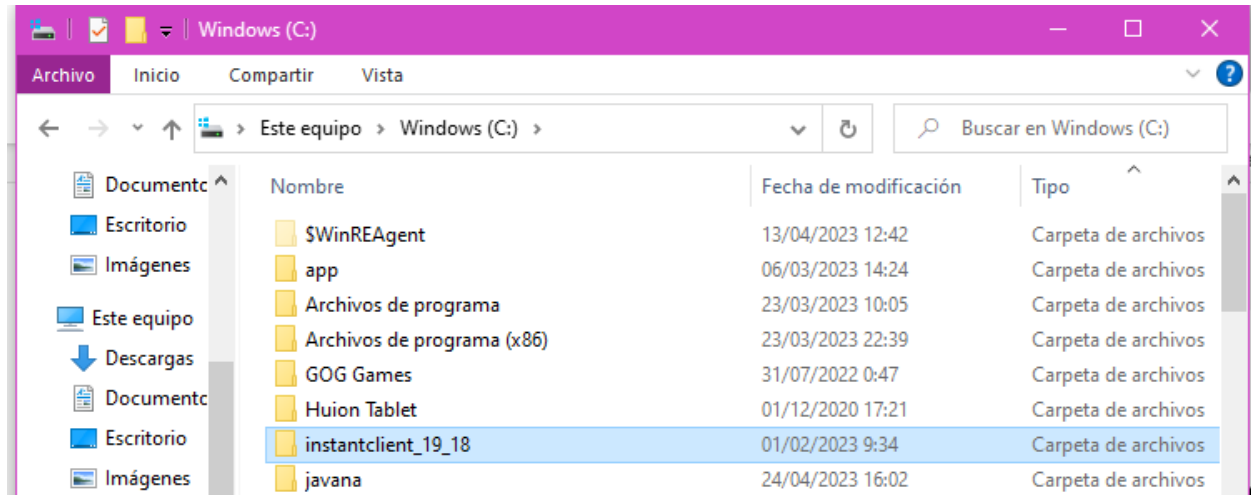
20. Buscar index.php, click derecho y dar a PHP Server:Serve Project.

21. Abrir servidor local.

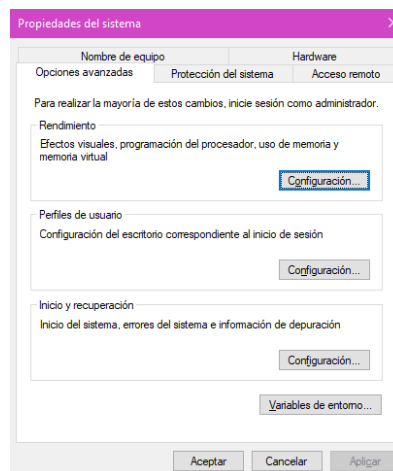
CONSEGUIDO!!

OCI8

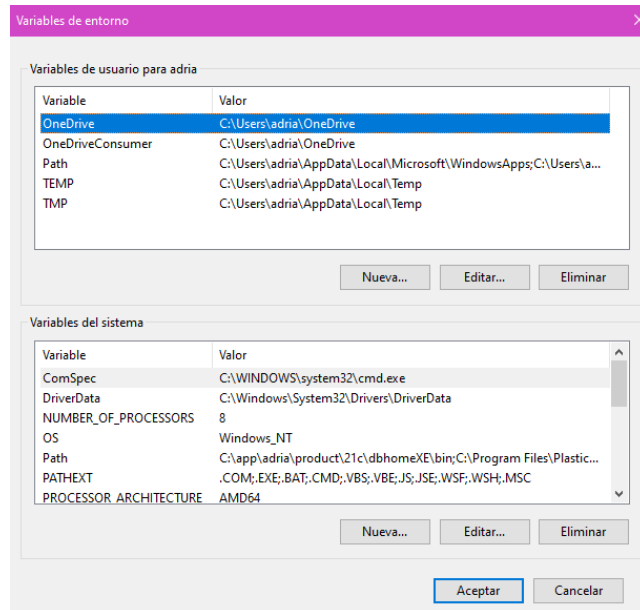
1. Acceder a <https://www.oracle.com/database/technologies/instant-client/downloads.html>
2. Selecciona tu sistema operativo.
3. Descargamos la versión compatible con nuestro PHP. En este proyecto, se ha utilizado la versión de PHP 8.2.0 y OCI8 19.18.
4. Se descomprime dentro de la raíz del sistema.



5. Buscamos en nuestro sistema “Variables de entorno”.
Se nos abrirá propiedades del sistema. Abrimos las variables de entorno.



6. Buscamos Path y le damos doble clic.



7. Añadimos instantclient dándole a Nuevo y añadiendo la ruta.

8. Buscamos en el sistema el archivo “php.ini”.

9. Accedemos a su ubicación.

10. Lo abrimos en un editor de texto (en nuestro caso Visual Studio Code).

11. Descomentamos la versión OCI8 que estemos utilizando. En nuestro caso, la 19.18.

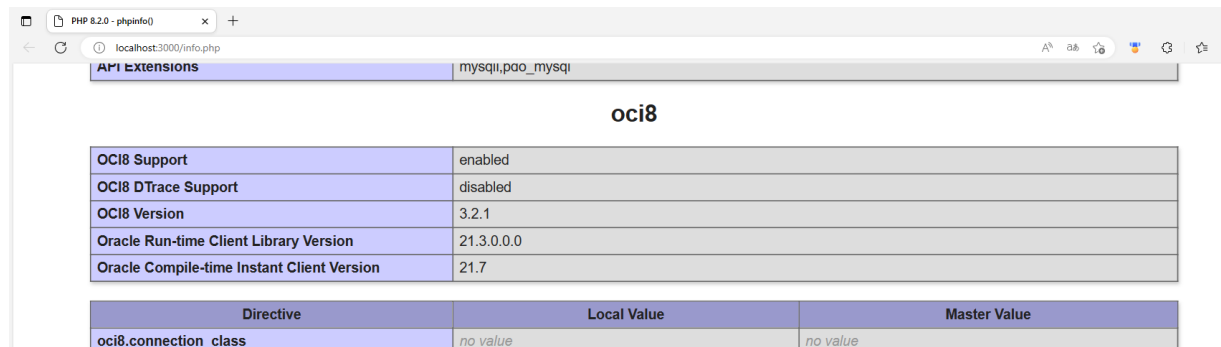
```
≡ php.ini ×
C: > xampp > php > ≡ php.ini
926 extension=mbstring
927 extension=exif ; Must be after mbstring as it depends on it
928 extension=mysqli
929 ;extension=oci8_12c ; Use with Oracle Database 12c Instant Client
930 extension=oci8_19 ; Use with Oracle Database 19 Instant Client
931 ;extension=odbc
932 ;extension=openssl
```

12. Reiniciamos el sistema.

13. Creamos un script php y lo ejecutamos.

```
info.php ×
info.php
1 <?php
2 phpinfo();
3
4 ?>
```

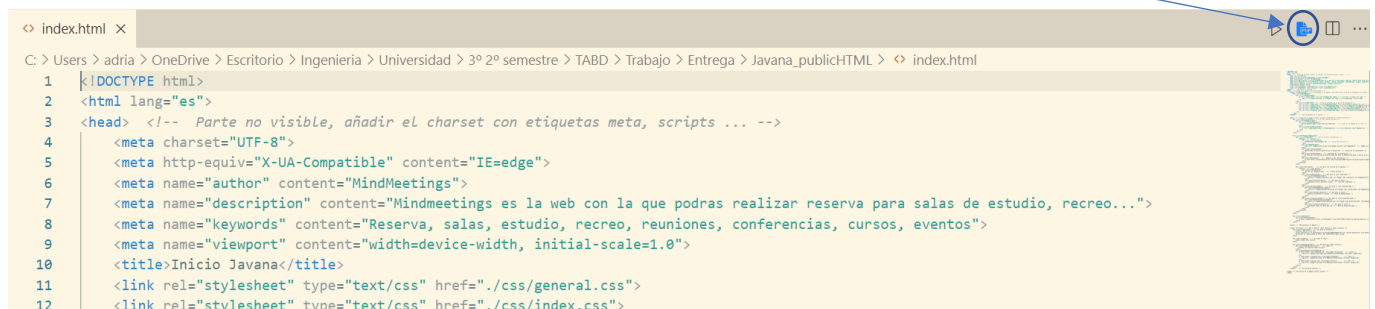
14. Nos saldrá una página en local host. Buscamos la sección oci8 y verificamos que en la línea “OCI8 Support” ponga enabled.



oci8	
OCI8 Support	enabled
OCI8 DTrace Support	disabled
OCI8 Version	3.2.1
Oracle Run-time Client Library Version	21.3.0.0.0
Oracle Compile-time Instant Client Version	21.7

Directive	Local Value	Master Value
oci8.connection_class	no value	no value

15. Accede desde VSCode al archivo “index.html” y pulsa el siguiente botón:



CONSEGUIDO!!

Referencias

- Affinity*. (2022). Obtenido de Estudio de abandono y adopción: <https://www.fundacion-affinity.org/observatorio/infografia-el-nunca-lo-haria-abandono-adopcion-perros-gatos-espana-2022>
- Archive, I. (s.f.). *WayBack Machine*. Obtenido de Database Awnsers: https://web.archive.org/web/20191120145428/http://databaseanswers.org/data_models/
- DataModeler y Ejemplo*. (2015). Obtenido de CodeGeek: <http://codigeek.blogspot.com/2015/04/sql-developer-data-modeler-y-ejemplo-de.html>
- Oracle OCI8*. (s.f.). Obtenido de <https://www.php.net/manual/es/book.oci8.php>
- Oracle Ya*. (s.f.). Obtenido de Tutoriales Ya: <https://www.tutorialesprogramacionya.com/oracleya/index.php?inicio=1>