Universidad de Castilla-La Mancha

Grado en Ingeniería Informática
Curso 2019/2020

Desarrollo de Bases de Datos

StartGrow

Yasín Muñoz El Merabety

yasin.munoz@alu.uclm.es

Javier Martínez Saltó

javier.martinez53@alu.uclm.es

Base de Datos: DBDD_25 QP0VCL9X

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

LISTA DE CAMBIOS

Nº Cambio	Fecha	Descripción
1	18/02/2020	Los usuarios de la base de datos Oracle de la portada han sido sustituidos por el usuario común que vamos a utilizar para la gestión de la misma.
2	22/02/2020	Se ha especificado con más detalle el Análisis de Requisitos según las dudas del profesor especificadas en los comentarios de retroalimentación de la entrega 1.
3	29/02/2020	Se ha actualizado el diagrama EER (Figura 1) con los atributos más significativos y se ha añadido las entidades Área y Proyecto.
4	02/03/2020	Se ha añadido el diagrama UML orientado a objetos.
5	12/03/2020	Se ha añadido el diseño lógico con los tipos, tablas y secuencias.
6	13/03/2020	Se ha añadido la implementación de la inserción de los datos.
7	14/03/2020	Se han añadido las vistas.
8	28/03/2020	Añadida la contraseña de la base de datos.
10	17/04/2020	Añadidas las consultas de Yasín.
11	18/04/2020	Añadidos los procedimientos de Yasín.
12	25/04/2020	Añadidos los *triggers* de Yasín.
13	25/04/2020	Diagramas MER y UML actualizados.
14	25/04/2020	Añadidas las consultas de Javier.
15	25/04/2020	Añadidos los procedimientos de Javier.
16	25/04/2020	Añadidos los *triggers* de Javier.
17	25/04/2020	Añadidos nuevos datos (nuevas solicitudes).
18	09/05/2020	Añadidas las órdenes de prueba para los procedimientos y *triggers* de Javier.
19	23/05/2020	Añadidas las órdenes de prueba para los *triggers* de Yasín.
20	23/05/2020	Añadido el apartado de gestión de datos XML.

1. ANÁLISIS DE REQUISITOS DEL SISTEMA.

La financiación es uno de los factores principales de éxito, y uno de los principales riesgos, a los que se enfrenta una startup para poder llevar a cabo una idea de negocio de carácter innovador. Ante la dificultad y la necesidad que tienen estas empresas emergentes para poder financiar y poner en marcha su modelo de negocio, nace StartGrow.

StartGrow es una plataforma web que facilita a los promotores, las startups, conseguir financiación a través de inversores que buscan una oportunidad de rentabilidad en estas empresas. Va dirigida a promotores con proyectos de carácter innovador y tecnológico, con un plan de negocio establecido, y a inversores que están interesados en apoyar con su inversión a este tipo de empresas.

El sistema está diseñado para soportar tres tipos de **usuarios**: empleado, promotor e inversor.

El **promotor** es el usuario que se registra en nuestra plataforma web para poder conseguir financiación para su startup. A la hora de registrarse deberá de indicar su información personal junto con un número de cuenta bancaria.

Una vez que el promotor se haya registrado en nuestra plataforma web, debe realizar una **solicitud**, que consiste en un formulario donde se pedirá datos básicos sobre la empresa.

El **empleado** es el encargado de gestionar una solicitud realizada por un promotor. Un empleado de StartGrow se pondrá en contacto con el promotor y estudiará dicha solicitud. En función de si cumple unos requisitos de viabilidad la podrá aceptar o rechazar. A la hora de contratar a un empleado, el administrador tendrá que registrarlo en el sistema indicando su información personal junto con su número de cuenta bancaria, puesto y salario.

Si el empleado aprueba la solicitud, se analizará toda la información de la startup, contactando con el promotor, gracias a nuestro equipo de analistas financiero. Se obtendrán datos útiles para el **inversor**, como el nivel de riesgo, rentabilidad, etc. A partir de este análisis, el empleado crea el **proyecto** de esta solicitud en el sistema y lo publica para que forme parte de nuestra plataforma web y los inversores empiecen a invertir en él.

La información que los inversores pueden ver del proyecto es el nombre de la startup, el área donde se desarrolla, el rating indicando el nivel de riesgo de la inversión, una estimación a cerca del plazo de retorno de la inversión, la cantidad mínima de inversión, la fecha límite para el fin de la ronda de financiación, el capital objetivo que necesita la startup, una estimación acerca de la rentabilidad esperada, el número de inversores y el progreso de la ronda de financiación.

Un proyecto puede estar en fase de financiación, financiado o cerrado. Un proyecto en fase de financiación es aquel en el cual se puede invertir. Los que están financiados son los que su ronda de financiación ha terminado, ya sea por límite de tiempo o porque ha conseguido la financiación que necesitaba. Los proyectos que están cerrados son aquellos en los cuales ya se han repartido todos los beneficios entre los inversores.

El **inversor** es el usuario que busca su oportunidad de **inversión**. Para poder invertir, antes debe de registrarse en nuestra plataforma web. A la hora de registrarse deberá de indicar su información personal. Después debe añadir una cantidad mínima de dinero en su cuenta, utilizando su tarjeta de crédito o cuenta bancaria. Una vez añadidos estos fondos, podrá invertir.

Cuando el inversor inicia sesión en nuestra plataforma web se le mostrará todos los proyectos disponibles para invertir, y aquellos que ya están financiados o cerrados. También podrá consultar todo tipo de información acerca de la evolución de sus inversiones en su *dashboard*: capital invertido y beneficios en un determinado rango de tiempo, capital disponible y cualquier detalle de cada inversión realizada. El inversor podrá retirar el dinero disponible en su cuenta sin ninguna restricción.

Una vez terminada la ronda de financiación del proyecto, los inversores podrán obtener el **retorno** de su inversión. Estos retornos se reparten entre los inversores de un proyecto dentro de los diez primeros días hábiles de cada mes, a mes vencido, durante un plazo establecido previamente.

2. DIAGRAMA ENTIDAD-RELACIÓN.

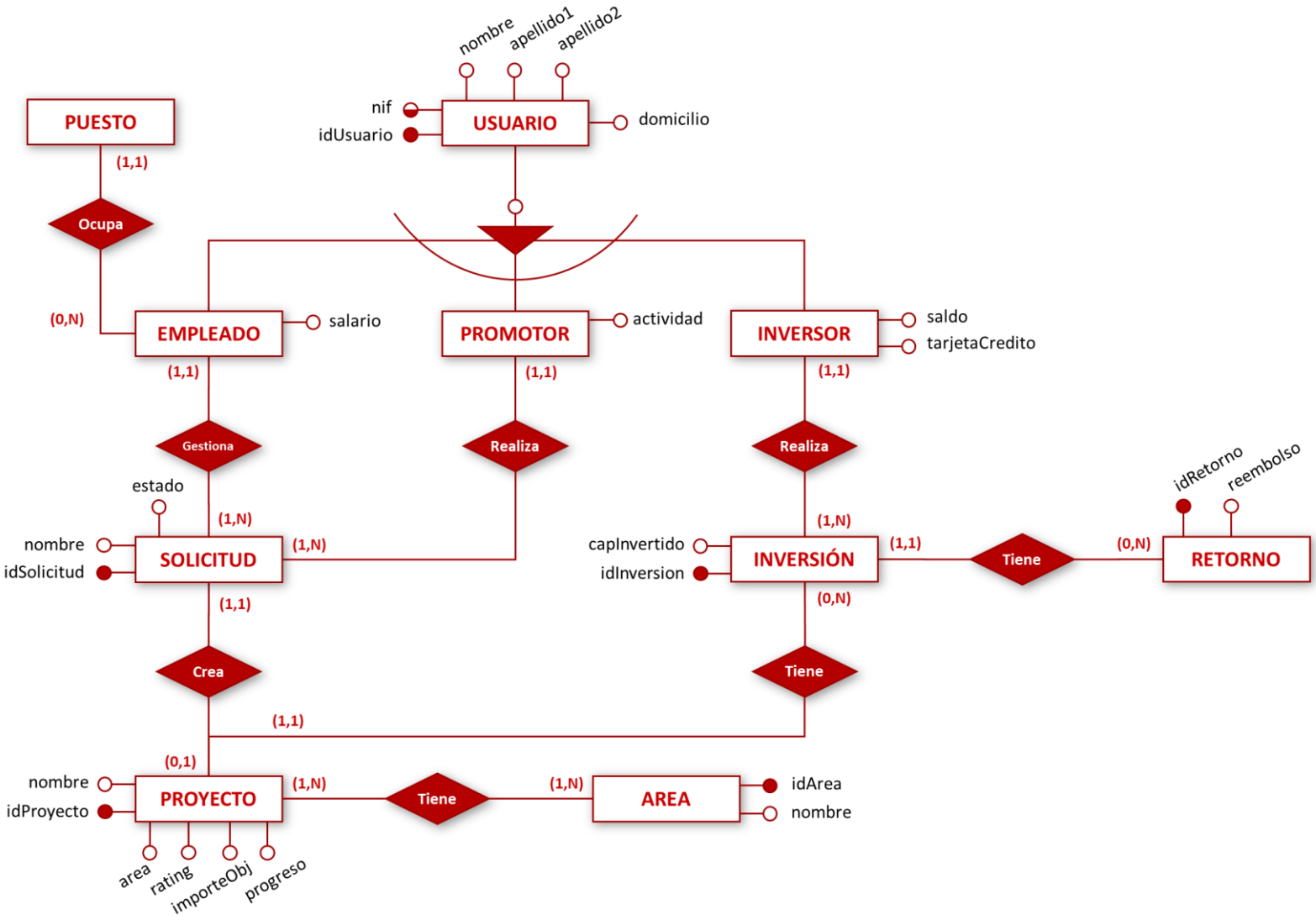

Figura 1: Diagrama Entidad-Relación StartGrow.

3. DIAGRAMA UML ORIENTADO A OBJETOS.

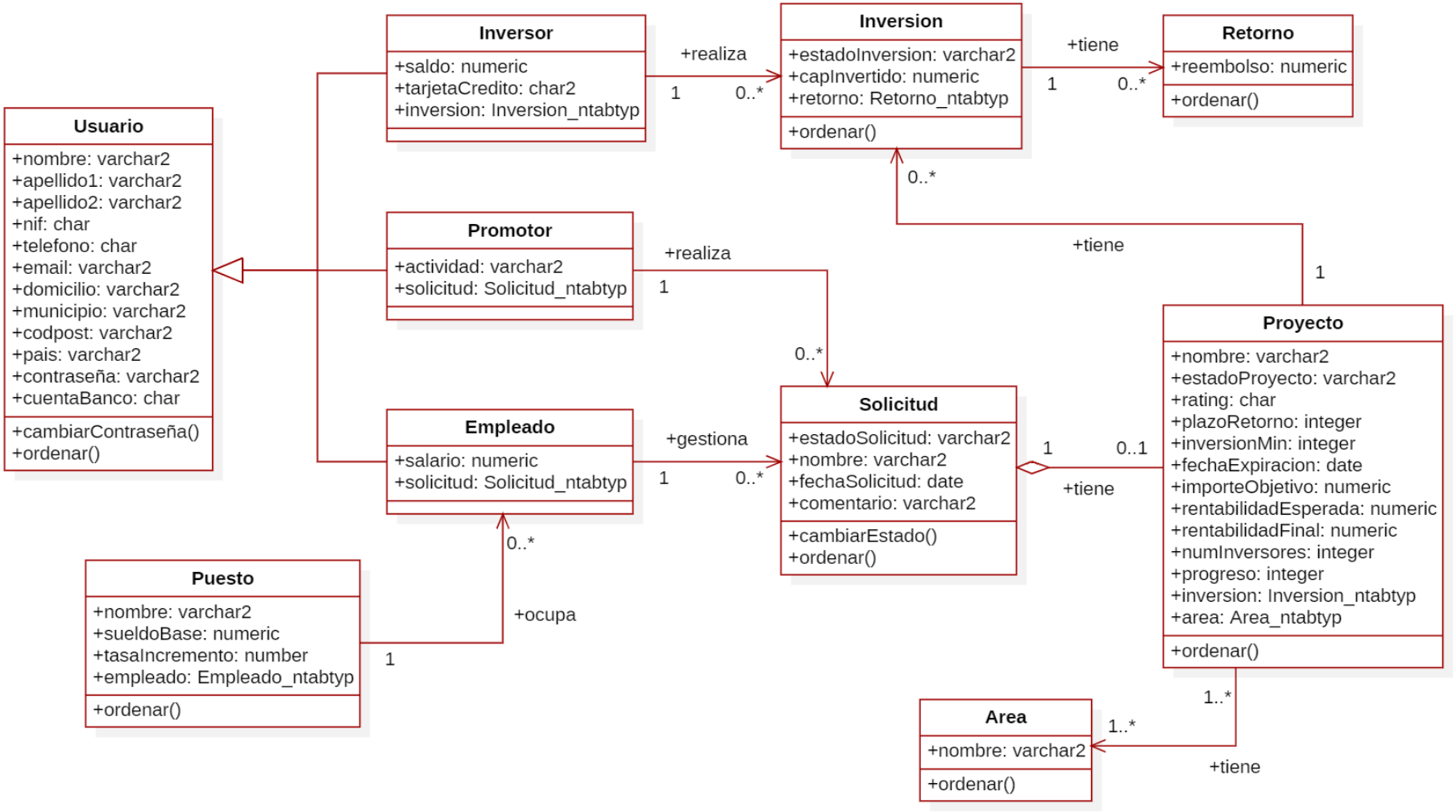

Figura 2: Diagrama UML StartGrow.

4. DISEÑO LÓGICO.

4.1. Tipos y Tablas.

```
-- Tipo USUARIO
create or replace type Usuario_objtyp as object (
    idUsuario number,
    nombre varchar2 (15),
    apellido1 varchar2 (20),
    apellido2 varchar2 (20),
    nif char (9),
    telefono char (9),
    email varchar2 (75),
    domicilio varchar2 (50),
    municipio varchar2 (20),
    codpost varchar2 (10),
    pais varchar2 (30),
    contraseña varchar2 (30),
    cuentaBanco char (24),

    member procedure cambiarContraseña  (nuevaContraseña in varchar2),
    order member function ordenar (v_usuario in Usuario_objtyp)
    return integer
) not INSTANTIABLE not FINAL;




/



-- Tabla USUARIO
create table Usuario_objtab of Usuario_objtyp (
    idUsuario primary key,
    check (apellido1 is not null),
    check (apellido2 is not null),
    nif unique,
    check (telefono is not null),
    check (email is not null),
    check (domicilio is not null),
    check (municipio is not null),
    check (codpost is not null),
    check (pais is not null),
    check (contraseña is not null),
    check (cuentaBanco is not null)
);

/
```

```
-- Función cambiarContraseña, ordenarUsuario
create or replace TYPE BODY usuario_objtyp AS
    MEMBER PROCEDURE cambiarContraseña(nuevaContraseña in VARCHAR2) IS
    BEGIN
        IF self IS OF (usuario_objtyp) THEN
            UPDATE usuario_objtab
            SET contraseña = nuevaContraseña
            WHERE idusuario = SELF.idusuario;
            DBMS_OUTPUT.PUT_LINE('La contraseña del usuario ha sido
            cambiada');
        end if;
    end;

    order member function ordenar (v_usuario in Usuario_objtyp)
    return integer is
    BEGIN
    if v_usuario.idUsuario < self.idUsuario then
        DBMS_OUTPUT.PUT_LINE('ID mayor.');
        return 1;
    else
        DBMS_OUTPUT.PUT_LINE('ID menor.');
        return 0;
    end if;
    END;
end;
/
-- Tipo SOLICITUD
create or replace type Solicitud_objtyp as object (
    idSolicitud number,
    estadoSolicitud varchar2 (15),
    nombre varchar2 (40),
    fechaSolicitud date,
    comentario varchar2 (500),

    member procedure cambiarEstado (estado in varchar2),
    order member function ordenar (v_solicitud in Solicitud_objtyp)
    return integer
);
/
-- Tabla anidada para EMPLEADO y PROMOTOR
create type Solicitud_ntabtyp as table of ref Solicitud_objtyp;
/
-- Tabla SOLICITUD
create table Solicitud_objtab of Solicitud_objtyp (
    idSolicitud primary key,
    check (UPPER(estadoSolicitud) in ('ACEPTADA','RECHAZADA','EN CURSO')),
    check (nombre is not null),
    check (fechaSolicitud is not null),
    check (comentario is not null)
);

/
```

```
create or replace type body Solicitud_objtyp as
    member procedure cambiarEstado (estado in varchar2) is
    BEGIN
      update Solicitud_objtab
      set estadoSolicitud = estado
      where idSolicitud = self.idSolicitud;
      DBMS_OUTPUT.PUT_LINE('Estado de la solicitud actualizado.');
    END;

    order member function ordenar (v_solicitud in Solicitud_objtyp)
    return integer is
    BEGIN
    if v_solicitud.idSolicitud < self.idSolicitud then
        DBMS_OUTPUT.PUT_LINE('ID mayor.');
        return 1;
    else
        DBMS_OUTPUT.PUT_LINE('ID menor.');
        return 0;
    end if;
    END;
END;

/

-- Tipo RETORNO
create type Retorno_objtyp as object (
    idRetorno number,
    reembolso numeric (7,2),

    order member function ordenar (v_retorno in Retorno_objtyp)
    return integer
);

/

create or replace type body Retorno_objtyp as
    order member function ordenar (v_retorno in Retorno_objtyp)
    return integer is
    BEGIN
    if v_retorno.idRetorno < self.idRetorno then
        DBMS_OUTPUT.PUT_LINE('ID mayor.');
        return 1;
    else
        DBMS_OUTPUT.PUT_LINE('ID menor.');
        return 0;
    end if;
    END;
END;


/
```

```
-- Tabla anidada para INVERSION
create type Retorno_ntabtyp as table of ref Retorno_objtyp;

/

-- Tabla RETORNO
create table Retorno_objtab of Retorno_objtyp (
    idRetorno primary key,
    check (reembolso > 0)
);


/


-- Tipo INVERSION
create or replace type Inversion_objtyp as object (
    idInversion number,
    estadoInversion varchar2 (20),
    capInvertido numeric (12,2),
    retorno Retorno_ntabtyp,

    order member function ordenar (v_inversion in Inversion_objtyp)
    return integer
);


/


create or replace type body Inversion_objtyp as
    order member function ordenar (v_inversion in Inversion_objtyp) return
integer is
    BEGIN
    if v_inversion.idInversion < self.idInversion then
        DBMS_OUTPUT.PUT_LINE('ID mayor.');
        return 1;
    else
        DBMS_OUTPUT.PUT_LINE('ID menor.');
        return 0;
    end if;
    END;
END;


/


-- Tabla anidada para INVERSOR
create type Inversion_ntabtyp as table of ref Inversion_objtyp;
```

```
/
-- Tabla INVERSION
create table Inversion_objtab of Inversion_objtyp (
    idInversion primary key,
    check (UPPER(estadoInversion) in ('RECAUDACIÓN', 'FINALIZADA',
    'EN CURSO')),
    check (capInvertido > 0)
)
nested table retorno store as Retornos_ntab;

/
alter table Retornos_ntab
add (scope for (column_value) is Retorno_objtab);

/

-- Tipo PROMOTOR
create or replace type Promotor_objtyp under Usuario_objtyp (
    actividad varchar2 (500),
    solicitud Solicitud_ntabtyp
);

/

-- Tipo Empleado
create or replace type Empleado_objtyp under Usuario_objtyp (
    salario numeric (6,2), -- al mes
    solicitud Solicitud_ntabtyp
);

/

-- Tabla anidada para PUESTO
create type Empleado_ntabtyp as table of ref Empleado_objtyp;

/

-- Tipo INVERSOR
create or replace type Inversor_objtyp under Usuario_objtyp (
    saldo numeric (12,2),
    tarjetaCredito char (16),
    inversion Inversion_ntabtyp
);

/
-- Tipo AREA
create or replace type Area_objtyp as object (
    idArea number,
    nombre varchar2 (20),

    order member function ordenar (v_area in Area_objtyp) return integer
);
```

```
/

create or replace type body Area_objtyp as
    order member function ordenar (v_area in Area_objtyp) return integer is
    BEGIN
    if v_area.idArea < self.idArea then
        DBMS_OUTPUT.PUT_LINE('ID mayor.');
        return 1;
    else
        DBMS_OUTPUT.PUT_LINE('ID menor.');
        return 0;
    end if;
    END;
END;

/

-- Tabla anidada para PROYECTO
create type Area_ntabtyp as table of ref Area_objtyp;

/

-- Tabla AREA
create table Area_objtab of Area_objtyp (
    idArea primary key,
    check (nombre is not null)
);

/

-- Tipo PROYECTO
create or replace type Proyecto_objtyp as object (
    idProyecto number,
    idSolicitud ref Solicitud_objtyp,
    nombre varchar2 (30),
    estadoProyecto varchar2 (20), --ENUM
    rating char (1), --ENUM
    plazoRetorno integer, -- en meses
    inversionMin integer,
    fechaExpiracion date,
    importeObjetivo numeric (12,2),
    rentabilidadEsperada numeric (4,2),
    rentabilidadFinal numeric (4,2),
    numInversores integer,
    progreso integer,
    inversion Inversion_ntabtyp,
    area Area_ntabtyp,

    order member function ordenar (v_proyecto in Proyecto_objtyp) return
integer
);
```

```
/
create or replace type body Proyecto_objtyp as
    order member function ordenar (v_proyecto in Proyecto_objtyp) return
integer is
    BEGIN
    if v_proyecto.idProyecto < self.idProyecto then
        DBMS_OUTPUT.PUT_LINE('ID mayor.');
        return 1;
    else
        DBMS_OUTPUT.PUT_LINE('ID menor.');
        return 0;
    end if;
    END;
END;


/


-- Tabla PROYECTO
create table Proyecto_objtab of Proyecto_objtyp (
    idProyecto primary key,
    idSolicitud references Solicitud_objtab,
    check (nombre is not null),
    check (UPPER(estadoProyecto) in ('FINANCIADO', 'CERRADO',
    'EN FINANCIACIÓN')),
    check (rating in ('A', 'B', 'C', 'D')),
    check (plazoRetorno > 0),
    check (inversionMin > 0),
    check (fechaExpiracion is not null),
    check (importeObjetivo > 0),
    check (rentabilidadEsperada > 0),
    check (numInversores >= 0),
    check (progreso >= 0)
)
nested table inversion store as InversionesP_ntab,
nested table area store as Areas_ntab;


/


alter table InversionesP_ntab
add (scope for (column_value) is Inversion_objtab);


/


alter table Areas_ntab
add (scope for (column_value) is Area_objtab);
/
```

```
-- Tipo PUESTO
create type Puesto_objtyp as object (
    idPuesto number,
    nombre varchar2 (20),
    sueldoBase numeric (12,2),
    tasaIncremento number,
    empleado Empleado_ntabtyp,

    order member function ordenar (v_puesto in Puesto_objtyp) return integer
);


/


create or replace type body Puesto_objtyp as
    order member function ordenar (v_puesto in Puesto_objtyp) return integer
is
    BEGIN
    if v_puesto.idPuesto < self.idPuesto then
        DBMS_OUTPUT.PUT_LINE('ID mayor.');
        return 1;
    else
        DBMS_OUTPUT.PUT_LINE('ID menor.');
        return 0;
    end if;
    END;
END;


/


-- Tabla PUESTO
create table Puesto_objtab of Puesto_objtyp (
    idPuesto primary key,
    check (nombre is not null),
    check (sueldoBase > 0),
    check (sueldoBase is not null),
    check (tasaIncremento > 0),
    check (tasaIncremento is not null)
)
nested table empleado store as Empleados_ntab;


/


alter table Empleados_ntab
add (scope for (column_value) is Usuario_objtab);
/
```

4.2. Secuencias.

```
CREATE SEQUENCE usuario_idusuario_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE solicitud_idsolicitud_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE proyecto_idproyecto_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE retorno_idretorno_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE inversion_idinversion_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE area_idarea_seq START WITH 1 INCREMENT BY 1;
CREATE SEQUENCE puesto_idpuesto_seq START WITH 1 INCREMENT BY 1;

/
```

5. DATOS.

```
-- Solicitudes
insert into Solicitud_objtab values (
    Solicitud_objtyp (
      solicitud_idsolicitud_seq.nextval, 'ACEPTADA', 'Virtual
      Indie','27/06/2019', 'Nuestro modelo de negocio en Virtual Indie se
      basa en fabricar gafas de realidad virtual. Nuestro objetivo es que
      gracias a este proyecto podamos realizar un proceso de fabricación más
      barato y que eso sea lo que nos diferencie de nuestra competencia en el
      mercado actual.'
    )
);

/

insert into Solicitud_objtab values (
    Solicitud_objtyp (
      solicitud_idsolicitud_seq.nextval, 'ACEPTADA', 'Electronics Medica
      SL', '19/02/2020', 'Hola, somos E-Medica. Necesitamos financiación por
      un valor de 500.000 euros en un plazo de un año. Para ello solicitamos
      poder recibirla con vosotros. Hemos adjuntado nuestro plan de negocio
      donde mostramos el capital del que disponemos y un plan de inversión
      explicando donde va a ir el dinero que solicitamos.'
    )
);

/

insert into Solicitud_objtab values (
    Solicitud_objtyp (
      solicitud_idsolicitud_seq.nextval, 'EN CURSO', 'Volava SA',
      '20/02/2020', 'Volava SA es una empresa que busca financiación para
      desarrollar un nuevo modelo innovador de helicopteros, que usen la
      energía eólica como su fuente de combustible. Aquí adjuntamos más
      detalles de esta tecnología para nuestros futuros inversores.'
    )
);

/

insert into Solicitud_objtab values (
    Solicitud_objtyp (
      solicitud_idsolicitud_seq.nextval, 'EN CURSO', 'Protos Education SA',
      '22/02/2020', 'Protos Education SA es una empresa que busca
      financiación para desarrollar una plataforma web educativa..'
    )
);

/
```

```
insert into Solicitud_objtab values (
    Solicitud_objtyp (
      solicitud_idsolicitud_seq.nextval, 'EN CURSO', 'SubSole Grocery SA',
      '24/02/2020', 'SubSole Grocery SA es una empresa que busca financiación
      para abrir un supermercado de comida orgánica.'
    )
);

/

insert into Solicitud_objtab values (
    Solicitud_objtyp (
      solicitud_idsolicitud_seq.nextval, 'EN CURSO', 'Green SA',
      '26/02/2020', 'Green SA es una empresa con un modelo de negocio
      Business-to-employee que se dedica a la consultoría sobre energía
      fotovoltáica.'
    )
);

/

insert into Solicitud_objtab values (
    Solicitud_objtyp (
      solicitud_idsolicitud_seq.nextval, 'EN CURSO', 'STC SA', '28/02/2020',
      'STC SA es una empresa de seguridad que necesita financiación.'
    )
);

/

insert into Solicitud_objtab values (
    Solicitud_objtyp (
      solicitud_idsolicitud_seq.nextval, 'EN CURSO', 'KLANTEC SA',
      '02/03/2020', 'KLANTEC SA es una empresa de ingeniería que necesita
      financiación para abrir una sucursal en Francia.'
    )
);

/

insert into Solicitud_objtab values (
    Solicitud_objtyp (
      solicitud_idsolicitud_seq.nextval, 'EN CURSO', 'Juguetería LMC SA',
      '04/03/2020', 'Juguetería LMC SA es una empresa de comercio que
      necesita financiación para vender sus juguetes en internet.'
    )
);

/
```

```
insert into Solicitud_objtab values (
    Solicitud_objtyp (
      solicitud_idsolicitud_seq.nextval, 'EN CURSO', 'Marcos Carretero
      Logistica SL', '06/03/2020', 'Marcos Carretero logísitca SL es una
      empresa de transportes que necesita financiación para la compra de
      nuevos camiones híbridos.'
    )
);

/

-- Usuarios Empleados
insert into Usuario_objtab values (
    Empleado_objtyp (
      usuario_idusuario_seq.nextval, 'Jesús', 'Buendía',
      'Martínez','47474747A', '684256875', 'jesus.buendia@startgrow.com',
      'C/La Almendra,26,1ºH', 'Albacete', '02001', 'España', '@1234',
      'ES3721001439539824252398', 1802.25, Solicitud_ntabtyp()
    )
);

/
insert into table (
    select treat (value(u) as Empleado_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 1
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 1)
);

/

insert into table (
    select treat (value(u) as Empleado_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 1
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 2)
);


/
```

```
insert into table (
    select treat (value(u) as Empleado_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 1
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 8)
);

/

insert into table (
    select treat (value(u) as Empleado_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 1
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 9)
);

/

insert into Usuario_objtab values (
    Empleado_objtyp (
      usuario_idusuario_seq.nextval, 'Damian', 'Sánchez', 'Torres',
      '12121212B', '784754855', 'damian.sanchez@startgrow.com', 'C/La
      Nuez,27,5ºA', 'Madrid', '28001', 'España', '@1235',
      'ES4204878989806946858819', 1965.78, Solicitud_ntabtyp()
    )
);


/

insert into table (
    select treat (value(u) as Empleado_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 2
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 3)
);

/
```

```
insert into table (
    select treat (value(u) as Empleado_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 2
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 4)
);

/

insert into table (
    select treat (value(u) as Empleado_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 2
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 5)
);

/

insert into Usuario_objtab values (
    Empleado_objtyp (
      usuario_idusuario_seq.nextval, 'José', 'García', 'Escribano',
      '37373737C', '604717593', 'jose.garcia@startgrow.com', 'C/La
      Madriguera,28,2ºC', 'Cáceres', '10005', 'España', '@1236',
      'ES9829489001242528783468', 2113.46, Solicitud_ntabtyp()
    )
);

/

insert into table (
    select treat (value(u) as Empleado_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 3
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 6)
);

/
```

```
insert into table (
    select treat (value(u) as Empleado_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 3
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 7)
);

/

insert into table (
    select treat (value(u) as Empleado_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 3
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 10)
);

/

-- Usuarios Promotores
insert into Usuario_objtab values (
    Promotor_objtyp (
      usuario_idusuario_seq.nextval, 'Florentino', 'Nieto', 'González',
      '27212721F', '679323593', 'florentino.nieto@hotmail.com', 'C/Via
      Trajana,51,4ºA', 'Barcelona', '08020', 'España', '@1239',
      'ES8937691103860937681045', 'Director Ejecutivo de Virtual Indie.',
      Solicitud_ntabtyp()
    )
);

/

insert into table (
    select treat (value(u) as Promotor_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 4
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 1)
);

/
```

```
insert into table (
    select treat (value(u) as Promotor_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 4
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 4)
);

/

insert into Usuario_objtab values (
    Promotor_objtyp (
      usuario_idusuario_seq.nextval,'Jose','López','Ruiz','13131313C','656743
      672', 'jose.lopez@hotmail.com','C/La Piña,2,2ºB', 'Valladolid',
      '47001','España','@1236', 'ES8104878129583112365513', 'Fundador de E-
      Medica junto con 3 socios.', Solicitud_ntabtyp()
    )
);

/

insert into table (
    select treat (value(u) as Promotor_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 5
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 2)
);

/


insert into Usuario_objtab values (
    Promotor_objtyp (
      usuario_idusuario_seq.nextval,'María','Peña','Domingo','14141414D',
      '705369712', 'maria.peña@gmail.com','C/La Alcachofa,5, 8ºA',
      'Salamanca','37001','España','@1237','ES8431909294989458517572','Direct
      or general de Volava. Mi función es dirigir la empresa y buscar fuentes
      de financiación.', Solicitud_ntabtyp()
    )
);

/
```

```
insert into table (
    select treat (value(u) as Promotor_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 6
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 3)
);


/

insert into Usuario_objtab values(
      Promotor_objtyp(
            usuario_idusuario_seq.nextval,'Carmen','González','Pérez','4756823
            9P','654782346','carmen.gonzalez@hotmail.com','C/Fresas,13','Palma
            de Mallorca','07001','España','@1523',
            'ES1401287843663179595464','Gerente de Subsole. Mi función es
            dirigir la empresa y buscar fuentes de financiación.',
            Solicitud_ntabtyp()
    )
);

/

insert into table (
    select treat (value(u) as Promotor_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 7
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 5)
);

/

insert into Usuario_objtab values (
      Promotor_objtyp (
      usuario_idusuario_seq.nextval,'Pablo','Serrano','López','15268743H','65
      9753852','pablo.serrano@hotmail.com','C/Papaya,13',
      'Ceuta','51001','España','@48563','ES3800752811858695525358','Gerente
      de Green. Mi función es dirigir la empresa y buscar fuentes de
      financiación.', Solicitud_ntabtyp()
    )
);

/
```

```
insert into table (
    select treat (value(u) as Promotor_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 8
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 6)
);

/


insert into Usuario_objtab values (
      Promotor_objtyp (
      usuario_idusuario_seq.nextval,'Marta','Pérez','Canuto','47461589M','789
      486153', 'marta.perez@hotmail.com','C/Mango,7', 'Ciudad
      Real','13001','España','@7703','ES0704875448082543597181','Gerente de
      STC. Mi función es dirigir la empresa y buscar fuentes de
      financiación.', Solicitud_ntabtyp()
    )
);

/

insert into table (
    select treat (value(u) as Promotor_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 9
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 7)
);

/

insert into Usuario_objtab values (
    Promotor_objtyp (
    usuario_idusuario_seq.nextval,'Carlos','Herrera','Gómez','12457863P','705
    869421','carlos.herrera@hotmail.com','C/Piña,72',
    'Madrid','28010','España','@6613','ES7621006394134282785264','Gerente de
    Klantec. Mi función es dirigir la empresa y buscar fuentes de
    financiación.', Solicitud_ntabtyp()
    )
);

/
```

```
insert into table (
    select treat (value(u) as Promotor_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 10
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 8)
);

/

insert into Usuario_objtab values (
    Promotor_objtyp (
    usuario_idusuario_seq.nextval,'Ana','Gómez','Ferreras','65413289K','64842
    3987','ana.gomez@hotmail.com','C/Pera,13','Madrid','28024','España','@135
    13','ES7220951879156521431895','Gerente de Juguetería LMC. Mi función es
    dirigir la empresa y buscar fuentes de financiación.',
    Solicitud_ntabtyp()
    )
);

/

insert into table (
    select treat (value(u) as Promotor_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 11
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 9)
);

/

insert into Usuario_objtab values (
      Promotor_objtyp(
      usuario_idusuario_seq.nextval,'Marcos','Carretero','Pereza','06487391Y'
      ,'729813465','marco.carretero@hotmail.com','C/Yogurt,14',
      'Barcelona','08007','España','@1813','ES3701287351631963911425','Gerent
      e de Marcos Carretero Logística SL. Mi función es dirigir la empresa y
      buscar fuentes de financiación.', Solicitud_ntabtyp()
    )
);

/
```

```
insert into table (
    select treat (value(u) as Promotor_objtyp).solicitud
    from Usuario_objtab u
    where idUsuario = 12
)
values (
    (select ref(s)
    from Solicitud_objtab s
    where idSolicitud = 10)
);

/
-- Retorno
-- De inversor 1, inversión 1
insert into Retorno_objtab values (
    Retorno_objtyp (
        retorno_idretorno_seq.nextval, 5000.00
    )
);

/

-- De inversor 1, inversión 2
insert into Retorno_objtab values (
    Retorno_objtyp (
        retorno_idretorno_seq.nextval, 1572.00
    )
);

/

-- De inversor 2, inversión 3
insert into Retorno_objtab values (
    Retorno_objtyp (
        retorno_idretorno_seq.nextval, 6000.00
    )
);

/


-- Inversión
-- De inversor 1, de proyecto finalizado
insert into Inversion_objtab values (
    Inversion_objtyp (
      inversion_idinversion_seq.nextval, 'RECAUDACIÓN', 11589.00,
      Retorno_ntabtyp()
    )
);

/
```

```
insert into table (
    select retorno
    from Inversion_objtab
    where idInversion = 1
)
values (
    (select ref(r)
    from Retorno_objtab r
    where idRetorno = 1)
);
/
-- De inversor 1, de proyecto finalizado
insert into Inversion_objtab values (
    Inversion_objtyp (
      inversion_idinversion_seq.nextval, 'RECAUDACIÓN', 1572.00,
      Retorno_ntabtyp()
    )
);

/
insert into table (
    select retorno
    from Inversion_objtab
    where idInversion = 2
)
values (
    (select ref(r)
    from Retorno_objtab r
    where idRetorno = 2)
);

/
-- De inversor 2, de proyecto finalizado
insert into Inversion_objtab values (
    Inversion_objtyp (
      inversion_idinversion_seq.nextval, 'RECAUDACIÓN', 9615.00,
      Retorno_ntabtyp()
    )
);

/
insert into table (
    select retorno
    from Inversion_objtab
    where idInversion = 3
)
values (
    (select ref(r)
    from Retorno_objtab r
    where idRetorno = 3)
);
/
```

```
-- De inversor 3, de proyecto finalizado
insert into Inversion_objtab values (
    Inversion_objtyp (
      inversion_idinversion_seq.nextval, 'RECAUDACIÓN', 7903.00,
      Retorno_ntabtyp()
    )
);

/
-- De inversor 3, de proyecto 2
insert into Inversion_objtab values (
    Inversion_objtyp (
      inversion_idinversion_seq.nextval, 'EN CURSO', 7205.00,
      Retorno_ntabtyp()
    )
);


/


-- Usuarios Inversor
insert into Usuario_objtab values (
    Inversor_objtyp (
        usuario_idusuario_seq.nextval, 'Lucía', 'Pérez', 'Muñoz',
        '15151515E', '676296378','lucia.perez@hotmail.com', 'C/La
        Fresa,12,2ºC', 'Palma de Mallorca', '07001', 'España', '@1238',
        'ES7100751186435136997493', 23000.00, '5508940724815660',
        Inversion_ntabtyp()
    )
);

/


insert into table (
    select treat (value(u) as Inversor_objtyp).inversion
    from Usuario_objtab u
    where idUsuario = 13
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 1)
);


/
```

```
insert into table (
    select treat (value(u) as Inversor_objtyp).inversion
    from Usuario_objtab u
    where idUsuario = 13
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 2)
);

/
insert into Usuario_objtab values (
    Inversor_objtyp (
      usuario_idusuario_seq.nextval,'Antonio', 'García', 'Martínez',
      '16161616F', '688856969', 'antonio.garcia@gmail.com', 'C/El
      Albaricoque,13,3D', 'Jaen', '23002 ', 'España', '@1239',
      'ES7020802567486791854281', 1568.78, '5559156876003462',
      Inversion_ntabtyp()
    )
);

/

insert into table (
    select treat (value(u) as Inversor_objtyp).inversion
    from Usuario_objtab u
    where idUsuario = 14
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 3)
);

/

insert into Usuario_objtab values (
    Inversor_objtyp (
      usuario_idusuario_seq.nextval,'Josefa', 'López', 'González',
      '17171717G', '659875421', 'josefa.lopez@hotmail.com', 'C/El
      Limón,17,4ºI', 'Sevilla', '41001', 'España', '@1230',
      'ES9520806487214562472181', 2015.15, '5421301422139099',
      Inversion_ntabtyp()
    )
);


/
```

```
insert into table (
    select treat (value(u) as Inversor_objtyp).inversion
    from Usuario_objtab u
    where idUsuario = 15
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 4)
);

/

insert into table (
    select treat (value(u) as Inversor_objtyp).inversion
    from Usuario_objtab u
    where idUsuario = 15
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 5)
);

/

insert into Usuario_objtab values (
    Inversor_objtyp (
        usuario_idusuario_seq.nextval, 'Ana', 'Navarro', 'Serrano',
        '18181818H', '744896245','ana.navarro@gmail.com', 'C/La Pera,5,1ºD',
        'Oviedo', '33001', 'España', '@1231', 'ES2531906198817219549159',
        2657.16, '5313682450839689', Inversion_ntabtyp()
    )
);

/

-- Area
insert into Area_objtab values (
    Area_objtyp (
        area_idarea_seq.nextval, 'TIC'
    )
);

/

insert into Area_objtab values (
    Area_objtyp (
        area_idarea_seq.nextval, 'Sanidad'
    )
);
/
```

```
insert into Area_objtab values (
    Area_objtyp (
        area_idarea_seq.nextval, 'Agricultura'
    )
);

/
insert into Area_objtab values (
    Area_objtyp (
        area_idarea_seq.nextval, 'Educación'
    )
);
/
insert into Area_objtab values (
    Area_objtyp (
        area_idarea_seq.nextval, 'Robótica'
    )
);

/
insert into Area_objtab values (
    Area_objtyp (
        area_idarea_seq.nextval, 'Cocina'
    )
);

/
insert into Area_objtab values (
    Area_objtyp (
        area_idarea_seq.nextval, 'Metalurgia'
    )
);

/
insert into Area_objtab values (
    Area_objtyp (
        area_idarea_seq.nextval, 'Transporte'
    )
);

/
insert into Area_objtab values (
    Area_objtyp (
        area_idarea_seq.nextval, 'Turismo'
    )
);
/
```

```
-- Proyecto
insert into Proyecto_objtab
    select proyecto_idproyecto_seq.nextval, ref(s), 'VirtualI', 'FINANCIADO',
    'C', 12, 26000.00, '12/02/2020', 35000.00, 7.00, 6.00, 3, 100,
    Inversion_ntabtyp(), Area_ntabtyp()
    from Solicitud_objtab s
    where s.idSolicitud = 1;

/


insert into table (
    select inversion
    from Proyecto_objtab
    where idProyecto = 1
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 1)
);

/

insert into table (
    select inversion
    from Proyecto_objtab
    where idProyecto = 1
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 2)
);

/

insert into table (
    select inversion
    from Proyecto_objtab
    where idProyecto = 1
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 3)
);


/
```

```
insert into table (
    select inversion
    from Proyecto_objtab
    where idProyecto = 1
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 4)
);

/
insert into table (
    select area
    from Proyecto_objtab
    where idProyecto = 1
)
values (
    (select ref(a)
    from Area_objtab a
    where idArea = 1)
);


/

insert into Proyecto_objtab
    select proyecto_idproyecto_seq.nextval, ref(s), 'E-MEDICA', 'EN
    FINANCIACIÓN', 'A', 12, 1000, '22/02/2021', 50000.00, 6.00, null, 0, 0,
    Inversion_ntabtyp(), Area_ntabtyp()
    from Solicitud_objtab s
    where s.idSolicitud = 2;


/


insert into table (
    select inversion
    from Proyecto_objtab
    where idProyecto = 2
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 5)
);


/
```

```
insert into table (
    select area
    from Proyecto_objtab
    where idProyecto = 2
)
values (
    (select ref(a)
    from Area_objtab a
    where idArea = 2)
);

/
-- Puesto
insert into Puesto_objtab values (
    Puesto_objtyp (
      puesto_idpuesto_seq.nextval, 'Gerente', 2000.00, 1, Empleado_ntabtyp()
    )
);

/

insert into table (
    select empleado
    from Puesto_objtab
    where idPuesto = 1
)
values (
    (select treat (ref(e) as ref Empleado_objtyp)
    from Usuario_objtab e
    where idUsuario = 3)
);

/

insert into Puesto_objtab values (
    Puesto_objtyp (
      puesto_idpuesto_seq.nextval, 'Administrativo', 1800.00, 1,
      Empleado_ntabtyp()
    )
);
/
insert into table (
    select empleado
    from Puesto_objtab
    where idPuesto = 2
)
values (
    (select treat (ref(e) as ref Empleado_objtyp)
    from Usuario_objtab e
    where idUsuario = 1)
);
/
```

```
insert into table (
    select empleado
    from Puesto_objtab
    where idPuesto = 2
)
values (
    (select treat (ref(e) as ref Empleado_objtyp)
    from Usuario_objtab e
    where idUsuario = 2)
);
```

6. CONSULTAS.

En este punto se exponen las consultas que hemos implementado cada uno de los desarrolladores de StartGrow sobre la base de datos, junto con la motivación y una breve explicación de la consulta.

6.1. Yasín.

6.1.1. Mostrar los empleados de StartGrow.

La vista muestra todos los datos de los empleados que trabajan en StartGrow. Como el tipo empleado_objtyp se trata de un subtipo de usuario_objtyp, debemos hacer un casting cada vez de queramos consultar la tabla. Por tanto, he considerado necesario implementar esta vista para que las consultas que realicemos sobre usuarios de tipo empleado sean más cortas y eficientes.

```
create view vista_empleado as
select t1.*,
       treat (value(t1) as empleado_objtyp).salario as salario,
       treat (value(t1) as empleado_objtyp).solicitud as solicitud
from usuario_objtab t1
where value(t1) is of (empleado_objtyp);
```

6.1.2. Mostrar todos los inversores de StartGrow.

La vista muestra todos los datos de los inversores que trabajan en StartGrow. Como el tipo inversor_objtyp se trata de un subtipo de usuario_objtyp, debemos hacer un casting cada vez de queramos consultar la tabla. Por tanto, he considerado necesario implementar esta vista para que las consultas que realicemos sobre usuarios de tipo inversor sean más cortas y eficientes.

```
create view vista_inversor as
select t1.*,
    treat (value(t1) as Inversor_objtyp).saldo as saldo,
    treat (value(t1) as Inversor_objtyp).tarjetacredito as tarjetacredito,
    treat (value(t1) as Inversor_objtyp).inversion as inversion
from Usuario_objtab t1
where value (t1) is of (Inversor_objtyp);
```

6.1.3. Mostrar todos los promotores de StartGrow.

La vista muestra todos los datos de los inversores que trabajan en StartGrow. Como el tipo promotor_objtyp se trata de un subtipo de usuario_objtyp, debemos hacer un casting cada vez de queramos consultar la tabla. Por tanto, he considerado necesario implementar esta vista para que las consultas que realicemos sobre usuarios de tipo promotor sean más cortas y eficientes.

```
create view vista_promotor as
select t1.*,
    treat (value(t1) as Promotor_objtyp).actividad as actividad,
    treat (value(t1) as Promotor_objtyp).solicitud as solicitud
from Usuario_objtab t1
where value (t1) is of (Promotor_objtyp);
```

6.1.4. Crear una solicitud por un promotor.

```
declare
    nombreEmpresa solicitud_objtab.nombre%type;
    comentario solicitud_objtab.comentario%type;
    usuario vista_promotor.idusuario%type;
    valorsecuencia solicitud_objtab.idsolicitud%type;
begin
    usuario := &Introduce_tu_id_usuario;
    nombreEmpresa := &Introduce_el_nombre_de_tu_empresa;
    comentario := &Introduce_un_comentario;

    insert into Solicitud_objtab values (
        Solicitud_objtyp (
            solicitud_idsolicitud_seq.nextval,
            null, nombreEmpresa, sysdate, comentario
        )
    );

    select solicitud_idsolicitud_seq.currval into valorsecuencia
    from dual;

    insert into table (
        select treat (value(u) as Promotor_objtyp).solicitud
        from Usuario_objtab u
        where idUsuario = usuario
    )
    values (
        (select ref(s)
        from Solicitud_objtab s
        where idSolicitud = valorsecuencia)
    );

    DBMS_OUTPUT.PUT_LINE('Muchas gracias por tu solicitud. Enseguida
contacteremos con usted.');

end;
```

6.1.5. Inversores que han invertido en proyectos de sanidad.

```
create view vista_inversores_sanidad as
select  t2.column_value.idInversion as idinversion,
        t2.column_value.capinvertido as CapitalInvertido,
        t1.idusuario, t1.nombre, t1.apellido1
from vista_inversor t1, table (t1.inversion) t2
inner join inversion_objtab t3
on t3.idinversion = t2.column_value.idInversion
inner join (select t5.column_value.idinversion as idinversion, t4.*
            from proyecto_objtab t4, table (t4.inversion) t5, table (t4.area)
            t6
            where t6.column_value.nombre = 'Sanidad') t7
on t2.column_value.idInversion = t7.idinversion;
```

6.1.6 Solicitudes en curso asignadas a cada empleado.

En StartGrow, a cada empleado se le asigna una serie de solicitudes para estudiar su aprobación. Sería útil consultar de vez en cuando cual es la carga de trabajo que tiene cada empleado. Para ello implementamos la siguiente vista:

```
create view solicitudesempleados as
select t2.column_value.idsolicitud as idsolicitud,
       t2.column_value.nombre as nombre,
       t1.nombre as "NOMBRE EMPLEADO",
       t1.idusuario as idusuario
from usuario_objtab t1, table(treat(value(t1) as empleado_objtyp).solicitud)
t2
where value(t1) is of (empleado_objtyp) and
      t2.column_value.estadosolicitud = 'EN CURSO'
order by idusuario;
```

6.1.7. Nombre, apellidos, teléfono y correo electrónico de los promotores cuyas solicitudes se estén estudiando.

Cuando recibimos una solicitud, nuestros empleados se ponen en contacto con el promotor para recoger información que nos permita estudiarla con detalle, por lo que necesitamos saber qué solicitudes son las que están en “En Curso”. Para ello implementamos la siguiente vista:

```
create view solicitudesencurso as
select t1.idusuario, t1.nombre,
       t1.apellido1, t1.apellido2,
       t1.telefono, t1.email,
       t2.column_value.nombre as "NOMBRE SOLICITUD"
from usuario_objtab t1, table(treat (value (t1) as
promotor_objtyp).solicitud) t2
where value (t1) is of (promotor_objtyp)
and   t2.column_value.estadosolicitud = 'EN CURSO';
```

6.1.8. Salario de los empleados con su nombre y apellidos.

```
create view salarioempleados as
    select treat (value (p) as Empleado_objtyp).nombre as nombre,
           treat (value (p) as Empleado_objtyp).apellido1 as apellido1,
           treat (value (p) as Empleado_objtyp).apellido2 as apellido2,
           treat (value (p) as Empleado_objtyp).salario as salario
    from usuario_objtab p
    where value (p) is of (empleado_objtyp);
```

6.1.9. Proyectos de menor riesgo y cuyo importe objetivo sea superior a 40000€.

```
create view proy_view1 as
    select *
    from proyecto_objtab
    where rating ='A' and importeobjetivo > 40000;
```

6.1.10. Inversores que tengan tarjeta de crédito y más de 2000€ de saldo.

```
Create view inv_view1 as
    select treat (value(p) as Inversor_objtyp).nombre as nombre,
       treat (value(p) as Inversor_objtyp).apellido1 as apellido1,
       treat (value(p) as Inversor_objtyp).apellido2 as apellido2
    from Usuario_objtab p
    where (treat (value (p) as Inversor_objtyp).saldo) > 2000
    and (treat (value (p) as Inversor_objtyp).tarjetaCredito) is not null;
```

6.2. Javier.

6.2.1. Mostrar las inversiones que tienen un proyecto.

```
CREATE VIEW proy_view2 as
    SELECT e.idProyecto, n.column_value
    FROM Proyecto_objtab e, table(e.inversion) n;
```

6.2.2. Mostrar los empleados que ocupan el puesto de administrativo.

```
CREATE VIEW puesto_view1 as
    SELECT p.idPuesto, n.column_value.nombre as nombre,
    n.column_value.apellido1 as apellido1,
    n.column_value.apellido2 as apellido2
    FROM Puesto_objtab p, table(p.empleado) n;
```

6.2.3. Mostrar las áreas de un proyecto financiado.

```
CREATE VIEW pareas_view1 as
    SELECT p.nombre, n.column_value.nombre as area
    FROM Proyecto_objtab p, table(p.area) n
    WHERE p.estadoProyecto = 'FINANCIADO';
```

6.2.4. Proyectos con su número de inversiones.

```
CREATE VIEW PROY_INV_VISTA AS
      SELECT IDPROYECTO, NOMBRE, ESTADOPROYECTO,
      COUNT(I.COLUMN_VALUE) AS NUM_INVERSIONES
      FROM PROYECTO_OBJTAB P, TABLE(P.INVERSION) I
      GROUP BY IDPROYECTO, NOMBRE, ESTADOPROYECTO;
```

6.2.5. Empleados con alguna solicitud asignada.

```
CREATE VIEW PROY_CONSOL_VIEW AS
      SELECT U.*, TREAT(VALUE(U) AS EMPLEADO_OBJTYP).SALARIO AS SALARIO,
      TREAT(VALUE(U) AS EMPLEADO_OBJTYP).SOLICITUD AS SOLICITUD
      FROM USUARIO_OBJTAB U
      WHERE VALUE(U) IS OF (EMPLEADO_OBJTYP) AND
      TREAT(VALUE(U) AS EMPLEADO_OBJTYP).SOLICITUD IS NOT NULL;
```

6.2.6. Solicitudes en curso y el empleado que las gestiona.

```
CREATE VIEW SOLEMP_ENCURSO_VIEW AS
SELECT S.*, E.IDUSUARIO, E.NOMBRE, E.APELLIDO1, E.APELLIDO2
FROM SOLICITUD_OBJTAB S, VISTA_EMPLEADO E, TABLE(E.SOLICITUD) SOL
WHERE S.ESTADOSOLICITUD = 'EN CURSO' AND SOL.COLUMN_VALUE = REF(S);
```

7. PROCEDIMIENTOS.

En este punto se exponen los procedimientos que hemos implementado cada uno de los desarrolladores de StartGrow sobre la base de datos, junto con la motivación y una breve explicación sobre su funcionamiento.

7.1. Yasín.

7.1.1. Comparación de salario.

El procedimiento compara el salario del empleado que le pasamos por parámetro con el salario de los demás empleados, mostrando por pantalla el nombre de aquellos empleados que cobran más que él. Puede servirnos para estudiar posibles aumentos salariales.

```
create or replace procedure comparasalario
  (empleado vista_empleado.idusuario%type)
is
    sal vista_empleado.salario%type;
    nom vista_empleado.nombre%type;
    ape vista_empleado.apellido1%type;
    nif vista_empleado.nif%type;

    cursor c1 (sal number) is
    select * from vista_empleado
    where salario > sal;
begin
    select nombre, apellido1, salario, nif
    into nom, ape, sal, nif
    from vista_empleado
    where idusuario = empleado;

    dbms_output.put_line (nom||' '||ape||' con nif '||nif||' cobra menos
    que:');

    for i in c1(sal) loop
        dbms_output.put_line(i.nombre||' '||i.apellido1||' con nif '||i.nif);
    end loop;

    exception
        when no_data_found then
        dbms_output.put_line ('No existe el empleado.');
end;
```

Para llamar al procedimiento ejecutamos el siguiente código:

```
set serveroutput on
begin
    comparasalario (/*idUsuario*/);
end;
```

7.1.2. Salario neto anual de un empleado.

En nuestra base de datos tenemos el salario mensual bruto de cada uno de nuestros empleados. Para saber el salario neto anual de un determinado empleado, implementamos el siguiente procedimiento:

```
create or replace procedure salarioneto
  (empleado vista_empleado.idusuario%type)
is
    sal number;
    nom vista_empleado.nombre%type;
begin
    select salario * 14,
           nombre
    into sal,nom
    from vista_empleado
    where idusuario = empleado;

    if (sal between 0 and 12450) then
        dbms_output.put_line('El sueldo neto de '||nom|| ' es '||(sal-
      sal*0.19)||' euros.');
    elsif (sal between 12451 and 20200) then
        dbms_output.put_line('El sueldo neto de '||nom|| ' es '||(sal-
      sal*0.24)||' euros.');
    elsif (sal between 20201 and 35201) then
        dbms_output.put_line('El sueldo neto de '||nom|| ' es '||(sal-
      sal*0.3)||' euros.');
    elsif (sal between 35201 and 60000) then
        dbms_output.put_line('El sueldo neto de '||nom|| ' es '||(sal-
      sal*0.37)||' euros.');
    elsif (sal > 60000) then
        dbms_output.put_line('El sueldo neto de '||nom|| ' es '||(sal-
      sal*0.45)||' euros.');
    end if;

    exception
        when no_data_found then
        dbms_output.put_line ('No existe el empleado.');

end;
```

Para llamar al procedimiento ejecutamos el siguiente código:

```
set serveroutput on
begin
    salarioneto (/*idusuario*/);
end;
```

7.1.3. Cantidad de dinero invertida por todos los inversores en un área.

La función devuelve un valor de tipo number con la cantidad de dinero invertida por los inversores en un área. Nos puede servir para saber en qué áreas son más atractivas para los inversores.

```
create or replace function totalInversionEnArea
   (nombrearea in area_objtab.nombre%type) return number
is
    total number;
begin
    select sum (t3.column_value.capinvertido) into total
    from proyecto_objtab t1, table (t1.area) t2, table (t1.inversion) t3
    where t2.column_value.nombre = nombrearea
    group by t2.column_value.nombre ;

    return total;

    exception
        when others then
            dbms_output.put_line ('No existe el área');
            return 0;
end;
```

Un ejemplo de uso de la función podría ser el de mostrar el total por pantalla llamando a la función de la siguiente manera:

```
set serveroutput on
begin
    dbms_output.put_line(totalInversionEnArea('TIC'));
end;
```

7.2. Javier.

7.2.1. Añadir cantidad al saldo de un Inversor.

Pasamos la cantidad que queremos añadir al saldo del inversor, además del ID del mismo para poder añadir la cantidad al usuario correspondiente.

```
CREATE OR REPLACE PROCEDURE ADDSALDO(CANTIDAD NUMERIC, IDI
USUARIO_OBJTAB.IDUSUARIO%TYPE) IS
    INVERSOR INVERSOR_OBJTYP;
    BEGIN
        IF(CANTIDAD < 0) THEN
            RAISE_APPLICATION_ERROR(-20100, 'La cantidad que ha introducido
no puede ser negativa.');
        END IF;

        SELECT TREAT(VALUE(U) AS INVERSOR_OBJTYP) INTO INVERSOR
        FROM USUARIO_OBJTAB U
        WHERE IDUSUARIO = IDI;

        INVERSOR.SALDO := INVERSOR.SALDO + CANTIDAD;

        UPDATE USUARIO_OBJTAB U
        SET VALUE(U) = INVERSOR
        WHERE IDUSUARIO = IDI;

        DBMS_OUTPUT.PUT_LINE('El saldo del usuario con ID:'||IDI||' ahora es
de '||INVERSOR.SALDO||'€.');

        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                DBMS_OUTPUT.PUT_LINE('El usuario que ha introducido no
existe');

    END;
```

Para probar el procedimiento:

```
SET SERVEROUTPUT ON
BEGIN
    ADDSALDO(100, 13);
END;
```

7.2.2. Restar cantidad al saldo de un inversor.

Al igual que en el procedimiento “addSaldo”, en este procedimiento restamos la cantidad deseada al saldo del inversor.

```
CREATE OR REPLACE PROCEDURE RESTARSALDO(CANTIDAD NUMERIC, IDI
USUARIO_OBJTAB.IDUSUARIO%TYPE) IS
    INVERSOR INVERSOR_OBJTYP;
    BEGIN
        IF(CANTIDAD < 0) THEN
            RAISE_APPLICATION_ERROR(-20100, 'La cantidad que ha introducido
no puede ser negativa.');
        END IF;

        SELECT TREAT(VALUE(U) AS INVERSOR_OBJTYP) INTO INVERSOR
        FROM USUARIO_OBJTAB U
        WHERE IDUSUARIO = IDI;

        IF(CANTIDAD > INVERSOR.SALDO) THEN
            RAISE_APPLICATION_ERROR(-20111, 'La cantidad que ha introducido
no puede ser mayor al saldo actual.');
        END IF;

        INVERSOR.SALDO := INVERSOR.SALDO - CANTIDAD;

        UPDATE USUARIO_OBJTAB U
        SET VALUE(U) = INVERSOR
        WHERE IDUSUARIO = IDI;

        DBMS_OUTPUT.PUT_LINE('El saldo del usuario con ID:'||IDI||' ahora es
de '||INVERSOR.SALDO||'€.');

        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                DBMS_OUTPUT.PUT_LINE('El usuario que ha introducido no
existe');

    END;
```

Para probar el procedimiento:

```
SET SERVEROUTPUT ON
BEGIN
    RESTARSALDO(30000, 13);
END;
```

7.2.3. Cambiar el puesto de un empleado.

Cambiar el puesto de un empleado, si el nuevo puesto es distinto al que ya tiene el empleado.

```
CREATE OR REPLACE PROCEDURE CAMBIAR_EMP_PUESTO(IDE
USUARIO_OBJTAB.IDUSUARIO%TYPE, NOM PUESTO_OBJTAB.NOMBRE%TYPE) IS
    PUESTO_NUEVO PUESTO_OBJTAB.IDPUESTO%TYPE;
    PUESTO_ACTUAL PUESTO_OBJTAB.IDPUESTO%TYPE;
    EMP REF USUARIO_OBJTYP;

    BEGIN
        SELECT IDPUESTO INTO PUESTO_NUEVO
        FROM PUESTO_OBJTAB
        WHERE NOMBRE = NOM;

        SELECT REF(E) INTO EMP
        FROM USUARIO_OBJTAB E
        WHERE IDUSUARIO = IDE;

        -- Buscar el empleado en PUESTO_OBJTAB para eliminarlo del puesto
        FOR I IN (SELECT IDPUESTO PU FROM PUESTO_OBJTAB P) LOOP
            FOR J IN(SELECT E.COLUMN_VALUE EM FROM PUESTO_OBJTAB P,
            TABLE(P.EMPLEADO) E WHERE IDPUESTO = I.PU) LOOP
                IF(EMP = J.EM) THEN
                    PUESTO_ACTUAL := I.PU;

                    IF(PUESTO_ACTUAL != PUESTO_NUEVO) THEN
                        DELETE FROM TABLE(SELECT EMPLEADO  FROM PUESTO_OBJTAB
                        WHERE IDPUESTO = PUESTO_ACTUAL)
                       WHERE COLUMN_VALUE = EMP;

                        INSERT INTO TABLE(
                            SELECT EMPLEADO
                            FROM PUESTO_OBJTAB
                            WHERE IDPUESTO = PUESTO_NUEVO
                        )
                        VALUES (
                            (SELECT TREAT(REF(E) AS REF EMPLEADO_OBJTYP)
                            FROM USUARIO_OBJTAB E
                            WHERE REF(E) = EMP)
                        );

                    ELSE
                        RAISE_APPLICATION_ERROR(-20200, 'El puesto actual y
                        el nuevo son el mismo, no ha pasado nada.');
                    END IF;
                ELSE
                        RAISE_APPLICATION_ERROR(-20220, 'El usuario que has
                        introducido no es un empleado.');
                END IF;
            END LOOP;
        END LOOP;
    END;
```

Para probar el procedimiento:

```
SET SERVEROUTPUT ON
BEGIN
    CAMBIAR_EMP_PUESTO(5, 'Gerente');
END;
```

8. BASE DE DATOS ACTIVA.

La base de datos de StartGrow necesita ejecutar de forma automática ciertas acciones cuando se producen determinados eventos bajo ciertas condiciones. Para ello cada desarrollador ha implementado los siguientes triggers.

8.1. Yasín.

8.1.1. Establecer el estado de solicitud "En Curso" cuando se inserta una solicitud.

```
create or replace trigger setSolicitudEnCurso
after insert on solicitud_objtab
declare
    valorsecuencia solicitud_objtab.idsolicitud%type;
begin
    select solicitud_idsolicitud_seq.currval into valorsecuencia
    from dual;

    update solicitud_objtab
    set estadosolicitud = 'EN CURSO'
    where idsolicitud = valorsecuencia;
end;
```

Para probar el *trigger* añadimos una solicitud al sistema. Lo podemos hacer ejecutando el siguiente código en pl/sql:

```
declare
    nombreEmpresa solicitud_objtab.nombre%type;
    comentario solicitud_objtab.comentario%type;

begin
    nombreEmpresa := &Introduce_el_nombre_de_tu_empresa;
    comentario := &Introduce_un_comentario;

    insert into Solicitud_objtab values (
        Solicitud_objtyp (
            solicitud_idsolicitud_seq.nextval,
            null, nombreEmpresa, sysdate, comentario
        )
    );

    DBMS_OUTPUT.PUT_LINE('Muchas gracias por tu solicitud. Enseguida
    contacteremos con usted.');

end;
```

Para consultar si la nueva solicitud se encuentra en el estado "EN CURSO", podemos simplemente hacer `select * from solicitud_objtab` para comprobarlo.

8.1.2. Establecer el estado de solicitud “Aceptada” y establecer datos base de un proyecto cuando se inserta un proyecto.

```
create or replace trigger setSolicitudAceptada
after insert on proyecto_objtab
declare
    valorsecuencia proyecto_objtab.idproyecto%type;
    valoridsolicitud solicitud_objtab.idsolicitud%type;
begin
    select proyecto_idproyecto_seq.currval into valorsecuencia
    from dual;

    select deref (idsolicitud).idsolicitud into valoridsolicitud
    from proyecto_objtab t1
    where t1.idproyecto = valorsecuencia;

    Update solicitud_objtab
    Set estadosolicitud = 'ACEPTADA'
    Where idsolicitud = valoridsolicitud;

    update proyecto_objtab
    set estadoproyecto = 'EN FINANCIACIÓN',
        rentabilidadfinal = null,
        numinversores = 0,
        progreso = 0
    where idproyecto = valorsecuencia;
end;
```

Para probar el *trigger* añadimos un proyecto al sistema. Por ejemplo, queremos aceptar la solicitud de la empresa de Volava SA (idSolicitud = 3) y por tanto subir un proyecto de financiación a StartGrow. Para crear el proyecto hacemos el siguiente insert:

```
select proyecto_idproyecto_seq.nextval, ref(s), 'Volava', null, 'A',
       16, 1200.00, '12/02/2021', 40000.00, 7.00, null, null, null,
       Inversion_ntabtyp(), Area_ntabtyp()
from Solicitud_objtab s
where s.idSolicitud = 3;
```

Después de insertar el proyecto, podemos hacer `select * from proyecto_objtab` para comprar que se han establecido los valores y podemos hacer `select * from solicitud_objtab` para comprobar que la solicitud queda registrada como aceptada después de haber subido el proyecto.

8.1.3. Promoción COVID-19.

StartGrow se compromete durante la pandemia en el apoyo a las startups que investigan en el campo de la salud. Para incentivar que nuestros inversores inviertan en este tipo de startups, StartGrow premiará con 1.000 euros a aquellos que hayan invertido más de 10.000 euros en el campo de la sanidad. Para ello, necesitamos implementar en nuestra base de datos el siguiente *trigger*:

```
create or replace trigger promocovid19
after logon on database
when (user ='DBDD_25') --DBDD_12
declare
    cursor c1 is
    select  sum (t2.column_value.capinvertido) as CapitalInvertido,
            t1.idusuario, t1.nombre
    from vista_inversor t1, table (t1.inversion) t2
    inner join inversion_objtab t3
    on t3.idinversion = t2.column_value.idInversion
    inner join (select t5.column_value.idinversion as idinversion,
                       t4.*
                from proyecto_objtab t4, table (t4.inversion) t5,
                table (t4.area) t6
                where t6.column_value.nombre = 'Sanidad') t7
    on t2.column_value.idInversion = t7.idinversion
    group by t1.idusuario, t1.nombre;
begin
    for i in c1 loop
        if i.capitalinvertido > 10000 then
            update vista_inversor
            set saldo = saldo + 1000
            where idusuario = i.idusuario;
        end if;
    end loop;
end;
```

Para probar el *trigger*, podemos hacer que Josefa (idUsuario = 15) invierta 4000 euros en E-Medica (idProyecto = 2), que es un proyecto del área de Sanidad. Como Josefa lleva invertido en esa área un total de 7205, si invierte otros 4000 euros, ya habrá superado la cantidad de los 10000 euros por lo que el trigger saltará y añadirá 1000 euros más a su saldo.

```
insert into Inversion_objtab values (
    Inversion_objtyp (
        inversion_idinversion_seq.nextval, 'EN CURSO', 4000.00,
        Retorno_ntabtyp()
    )
);
```

```
insert into table (
    select inversion
    from Proyecto_objtab
    where idProyecto = 2
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 6)
);

insert into table (
    select treat (value(u) as Inversor_objtyp).inversion
    from Usuario_objtab u
    where idUsuario = 15
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 6)
);
```

Antes de la inversión, Josefa tenía 2015,15 euros, y consultando su saldo haciendo

```
select nombre, saldo
from vista_inversor
where idusuario = 15;
```

podemos ver como ahora tiene 3015,15 euros. Como el trigger es after logon on, estos cambios tendrán efectos al reiniciar la sesión en el sistema.

8.2. Javier.

8.2.1. Asignar una solicitud que se acaba de crear a un empleado.

Cuando creamos una nueva solicitud, debe ser asignada a un empleado para que sea evaluada. Esta se asignará al empleado que menos solicitudes haya gestionado, debemos implementar el siguiente *trigger*:

```
CREATE OR REPLACE TRIGGER ASIGNAR_SOLICITUD
FOR INSERT ON SOLICITUD_OBJTAB
COMPOUND TRIGGER
    TYPE TIDSOLICITUD IS TABLE OF SOLICITUD_OBJTAB.IDSOLICITUD%TYPE
        INDEX BY BINARY_INTEGER;
    VTIDSOLICITUD TIDSOLICITUD;

    NE BINARY_INTEGER := 0; -- ÍNDICE DE VTIDSOLICITUD
    IDEMP BINARY_INTEGER := 0; -- EMPLEADO AL QUE SE ASIGNARÁ LA SOLICITUD
    NEMP BINARY_INTEGER := 0; -- VARIABLE AUX PARA CAMBIAR IDEMP
    NSOLEMP BINARY_INTEGER := 0; -- SOLICITUDES DEL EMPLEADO CON ID NEMP
    NSOL BINARY_INTEGER := 0; -- VARIABLE AUX PARA CAMBIAR NSOLEMP

BEFORE EACH ROW IS
BEGIN
    NE := NE + 1;
    VTIDSOLICITUD(NE) := :NEW.IDSOLICITUD;
END BEFORE EACH ROW;

AFTER STATEMENT IS
BEGIN
    SELECT COUNT(*) INTO NEMP
    FROM USUARIO_OBJTAB U
    WHERE VALUE(U) IS OF (EMPLEADO_OBJTYP);

    IF(NEMP > 0) THEN
        SELECT IDUSUARIO INTO IDEMP
        FROM USUARIO_OBJTAB U
        WHERE VALUE(U) IS OF (EMPLEADO_OBJTYP)
        FETCH FIRST 1 ROWS ONLY;

        SELECT COUNT(S.COLUMN_VALUE) INTO NSOLEMP
        FROM USUARIO_OBJTAB U, TABLE(TREAT(VALUE(U) AS
EMPLEADO_OBJTYP).SOLICITUD) S
        WHERE IDUSUARIO = IDEMP;

        FOR J IN 1..NE LOOP
            FOR i IN (SELECT IDUSUARIO IDU FROM USUARIO_OBJTAB U WHERE
VALUE(U) IS OF (EMPLEADO_OBJTYP)) LOOP
                SELECT COUNT(S.COLUMN_VALUE) INTO NSOL
                FROM USUARIO_OBJTAB U, TABLE(TREAT(VALUE(U) AS
EMPLEADO_OBJTYP).SOLICITUD) S
                WHERE IDUSUARIO = i.IDU;

                IF (NSOL < NSOLEMP) THEN
                    IDEMP := i.IDU;
```

```
                    NSOLEMP := NSOL;
                END IF;
            END LOOP;

            INSERT INTO TABLE (
                SELECT TREAT(VALUE(U) AS EMPLEADO_OBJTYP).SOLICITUD
                FROM USUARIO_OBJTAB U
                WHERE IDUSUARIO = IDEMP
            )
            VALUES (
                (SELECT REF(S)
                FROM SOLICITUD_OBJTAB S
                WHERE IDSOLICITUD = VTIDSOLICITUD(J))
            );

        END LOOP;

    ELSE
        DBMS_OUTPUT.PUT_LINE('No hay empleados a los que asignarles esta
solicitud.');
    END IF;

END AFTER STATEMENT;
END;
```

Para probar el *trigger*:

```
insert into Solicitud_objtab values (
    Solicitud_objtyp (
      solicitud_idsolicitud_seq.nextval, 'ACEPTADA', 'Virtual Indie',
      '27/06/2019', 'Nuestro modelo de negocio en Virtual Indie se basa en
      fabricar gafas de realidad virtual. Nuestro objetivo es que gracias a
      este proyecto podamos realizar un proceso de fabricación más barato y
      que eso sea lo que nos diferencie de nuestra competencia en el mercado
      actual.'
    )
);
```

8.2.2. Saldo mayor que cero al insertar o actualizar un empleado.

Necesitamos controlar el salario de nuestros empleados. Al insertar o actualizar uno nuevo, el salario debe ser siempre mayor que 0. Debemos implementar el siguiente *trigger*:

```
CREATE OR REPLACE TRIGGER CONSISTENCIA_SALARIO
FOR INSERT OR UPDATE ON USUARIO_OBJTAB
COMPOUND TRIGGER
    TYPE TIDUSUARIO IS TABLE OF USUARIO_OBJTAB.IDUSUARIO%TYPE
        INDEX BY BINARY_INTEGER;
    VTIDUSUARIO TIDUSUARIO;

    EMPLEADO USUARIO_OBJTYP;
    INDICE BINARY_INTEGER := 0; -- ÍNDICE DE VTIDUSUARIO
    VEMP USUARIO_OBJTYP; -- EL EMPLEADO
    SAL NUMERIC(6, 2); -- EL SALARIO DEL EMPLEADO

BEFORE EACH ROW IS
BEGIN
    INDICE := INDICE + 1;
    VTIDUSUARIO(INDICE) := :NEW.IDUSUARIO;
END BEFORE EACH ROW;

AFTER STATEMENT IS
BEGIN
    FOR I IN 1..INDICE LOOP
        SELECT VALUE(U) INTO EMPLEADO
        FROM USUARIO_OBJTAB U
        WHERE IDUSUARIO = VTIDUSUARIO(INDICE);

        IF EMPLEADO IS OF (EMPLEADO_OBJTYP) THEN
            SELECT VALUE(U), TREAT(VALUE(U) AS EMPLEADO_OBJTYP).SALARIO INTO
VEMP, SAL
            FROM USUARIO_OBJTAB U
            WHERE IDUSUARIO = VTIDUSUARIO(I) AND VALUE(U) IS OF
(EMPLEADO_OBJTYP);

            IF(SAL <= 0) THEN
                RAISE_APPLICATION_ERROR(-20101, 'El salario que ha
introducido debe ser mayor que 0.');
            END IF;

        END IF;
    END LOOP;

END AFTER STATEMENT;
END;
```

Para probar el *trigger*:

```
DECLARE
    E EMPLEADO_OBJTYP;
BEGIN
    SELECT TREAT(VALUE(U) AS EMPLEADO_OBJTYP) INTO E FROM USUARIO_OBJTAB U
    WHERE IDUSUARIO = 3;
    E.SALARIO := 0;

    UPDATE USUARIO_OBJTAB U SET VALUE(U) = E WHERE IDUSUARIO = 3;
END;
```

8.2.3. Una inversión no debe aparecer en más de un proyecto.

Debemos controlar que una inversión no se encuentra en ningún otro proyecto. Para ello comprobamos las inversiones de cada proyecto, así sabremos si podemos insertar o actualizar la nueva inversión.

Para poder crear este trigger debemos tener la siguiente vista:

```
CREATE VIEW INVERSION_PROY_VIEW AS
    SELECT *
    FROM PROYECTO_OBJTAB;
```

Ya podemos crear el trigger.

```
CREATE OR REPLACE TRIGGER INTEGRIDAD_INV_PROY
INSTEAD OF INSERT OR UPDATE ON NESTED TABLE INVERSION OF INVERSION_PROY_VIEW
FOR EACH ROW
DECLARE
  INV REF INVERSION_OBJTYP;
BEGIN
  INV := :NEW.COLUMN_VALUE;

  IF(INV IS NOT NULL) THEN
    IF UPDATING THEN
      IF (:NEW.COLUMN_VALUE = :OLD.COLUMN_VALUE) THEN
        RAISE_APPLICATION_ERROR(-20110, 'La inversión anterior y la nueva son la misma. Nada ha
cambiado.');
      END IF;
    END IF;

    FOR i IN (SELECT IDPROYECTO PROY FROM PROYECTO_OBJTAB P) LOOP
      FOR j IN (SELECT N.COLUMN_VALUE D FROM INVERSION_PROY_VIEW PR, TABLE(PR.INVERSION) N
WHERE IDPROYECTO = i.PROY) LOOP
        IF (INV = j.D) THEN
          RAISE_APPLICATION_ERROR(-20100, 'No se puede agregar esta inversión, ya existe en un
proyecto.');
        END IF;
      END LOOP;
    END LOOP;

    IF INSERTING THEN
      INSERT INTO TABLE (
        SELECT INVERSION
        FROM PROYECTO_OBJTAB
```

```
        WHERE IDPROYECTO = :PARENT.IDPROYECTO
      )
      VALUES (:NEW.COLUMN_VALUE);
    END IF;

    IF UPDATING THEN
      UPDATE TABLE (SELECT INVERSION FROM PROYECTO_OBJTAB WHERE IDPROYECTO =
:PARENT.IDPROYECTO)
      SET COLUMN_VALUE = :NEW.COLUMN_VALUE
      WHERE COLUMN_VALUE = :OLD.COLUMN_VALUE;
    END IF;

  ELSE
    RAISE_APPLICATION_ERROR(-20120, 'La nueva inversión es NULL. No contiene nada.');
  END IF;
END;
```

Para probar el *trigger*:

```
-- Update
UPDATE TABLE (SELECT INVERSION FROM INVERSION_PROY_VIEW WHERE IDPROYECTO = 1)
T
SET COLUMN_VALUE =
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 3)
WHERE T.COLUMN_VALUE.IDINVERSION = 4;

-- Select
insert into table (
    select inversion
    from INVERSION_PROY_VIEW
    where idProyecto = 1
)
values (
    (select ref(i)
    from Inversion_objtab i
    where idInversion = 4)
);
```

9. GESTIÓN DE DATOS XML.

Se ha decidido añadir para StartGrow una nueva funcionalidad que permita a los inversores, que estén registrados en nuestro sistema, la posibilidad de realizar una donación sobre cualquier proyecto que se deseé. En este tipo de transacción, como el término indica, no se permitirá ningún tipo de retorno para el inversor que la realice.

9.1. XML Schema.

A continuación, se indica el XML Schema que vamos a utilizar para validar los datos que se introduzcan. Si analizamos su contenido, podemos destacar que existe un tipo “TipoDonacion” para la etiqueta raíz del xml. Este contiene a su vez una secuencia con otros tres tipos de elementos, el proyecto, el inversor que hace la donación, y el donativo en sí.

Además, tenemos restricciones que los datos deben cumplir. El DNI del inversor será la clave primaria para ese tipo, que estará referenciada en el DNI del donativo. Esto quiere decir que el DNI que aparezca en el donativo debe existir en el inversor. Y, por último, el nombre de cada uno de los proyectos (“nombre_p”) será único en el sistema.

```
BEGIN
DBMS_XMLSCHEMA.REGISTERSCHEMA(SCHEMAURL=>'donacion.xsd',
schemadoc=> '<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="donacion" type="TipoDonacion">
    <xs:key name="ClaveDNI">
        <xs:selector xpath= "xs:inversor"/>
        <xs:field xpath="xs:dni"/>
    </xs:key>
    <xs:unique name="UProyecto">
        <xs:selector xpath= "xs:proyecto"/>
        <xs:field xpath="xs:nombre_p"/>
    </xs:unique>
    <xs:keyref name="RefInversor" refer="ClaveDNI">
        <xs:selector xpath="xs:donativo"/>
        <xs:field xpath="xs:dni"/>
    </xs:keyref>
</xs:element>

<xs:complexType name="TipoDonacion">
    <xs:sequence>
      <xs:element name="proyecto" type="TipoProyecto" maxOccurs="unbounded"/>
      <xs:element name="inversor" type="TipoInversor" maxOccurs="unbounded"/>
      <xs:element name="donativo" type="TipoDonativo" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:simpleType name="listAreas">
      <xs:list itemType="xs:string"/>
</xs:simpleType>
```

```
<xs:complexType name="TipoProyecto">
      <xs:sequence>
            <xs:element name="nombre_p" type="xs:string"/>
            <xs:element name="rating" type="xs:string"/>
            <xs:element name="area" type="listAreas"/>
      </xs:sequence>
      <xs:attribute name="estado" use="required" type="xs:string"/>
</xs:complexType>

<xs:complexType name="TipoInversor">
      <xs:sequence>
            <xs:element name="nombre_i" type="xs:string"/>
            <xs:element name="dni" type="xs:string"/>
            <xs:element name="saldo" type="xs:decimal"/>
      </xs:sequence>
</xs:complexType>

<xs:complexType name="TipoDonativo">
      <xs:sequence>
            <xs:element name="nombre_p" type="xs:string"/>
            <xs:element name="dni" type="xs:string"/>
            <xs:element name="cantidad" type="xs:decimal"/>
      </xs:sequence>
</xs:complexType>

</xs:schema>',

local=>true, gentypes=>false, genbean=>false, gentables=>false, force=>false,
options=>dbms_xmlschema.register_binaryxml, owner=>user);

commit;
END;
```

9.2. Creación de la Tabla Relacional.

```
CREATE TABLE DONACION_TAB (
      ID NUMBER PRIMARY KEY,
      DONACION XMLTYPE
)
XMLTYPE COLUMN DONACION STORE AS BINARY XML
XMLSCHEMA "donacion.xsd" ELEMENT "donacion";
```

9.3. Inserción de los datos.

```
INSERT INTO DONACION_TAB VALUES(1,
'<?xml version="1.0" encoding="UTF-8"?>
<donacion xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <proyecto estado="en financiacion">
            <nombre_p>ReBur</nombre_p>
        <rating>A</rating>
        <area>Agricultura Turismo</area>
      </proyecto>
      <proyecto estado="financiado">
            <nombre_p>Circle Zero</nombre_p>
        <rating>B</rating>
        <area>Robotica</area>
      </proyecto>

      <inversor>
            <nombre_i>Lucia Perez</nombre_i>
        <dni>15151515E</dni>
        <saldo>23000</saldo>
      </inversor>
      <inversor>
            <nombre_i>Antonio Garcia</nombre_i>
        <dni>16161616F</dni>
        <saldo>1568</saldo>
      </inversor>

      <donativo>
            <nombre_p>ReBur</nombre_p>
        <dni>15151515E</dni>
            <cantidad>10000</cantidad>
      </donativo>
      <donativo>
            <nombre_p>Circle Zero</nombre_p>
        <dni>16161616F</dni>
            <cantidad>500</cantidad>
      </donativo>
</donacion>'
);
```

```
INSERT INTO DONACION_TAB VALUES(2,
'<?xml version="1.0" encoding="UTF-8"?>
<donacion xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <proyecto estado="en financiacion">
        <nombre_p>ALVE</nombre_p>
        <rating>C</rating>
        <area>TIC</area>
      </proyecto>
      <proyecto estado="en financiacion">
        <nombre_p>VAINSA</nombre_p>
        <rating>B</rating>
        <area>Cocina</area>
      </proyecto>

      <inversor>
        <nombre_i>Diego Rodríguez</nombre_i>
        <dni>99994568Y</dni>
        <saldo>20000</saldo>
      </inversor>
      <inversor>
        <nombre_i>Sergio Martínez</nombre_i>
        <dni>18194574J</dni>
        <saldo>15000</saldo>
      </inversor>

      <donativo>
        <nombre_p>ALVE</nombre_p>
        <dni>99994568Y</dni>
        <cantidad>200</cantidad>
      </donativo>
      <donativo>
        <nombre_p>VAINSA</nombre_p>
        <dni>18194574J</dni>
        <cantidad>150</cantidad>
      </donativo>
</donacion>');
```

9.4. Indexación de la Tabla.

```
CREATE INDEX IDX_DONACION ON DONACION_TAB(donacion)
INDEXTYPE IS XDB.XMLINDEX;
```

9.5. Consultas XPath.

9.5.1. Yasín.

- Consulta 1: Nombre de los proyectos donde Sergio Martínez ha hecho un donativo.

```
CREATE VIEW DONATIVOS_SERGIO AS
      SELECT EXTRACT(DONACION, '//donativo [dni = //inversor [nombre_i =
      "Sergio Martínez"]/dni/text()]/nombre_p/text()') AS PROYECTO
FROM DONACION_TAB
WHERE ID = 2;
```

- Consulta 2: Nombre de los proyectos que han recibido más de 100 euros de donativo.

```
CREATE VIEW DONATIVOS_MAYORES_100 AS
      SELECT EXTRACT(DONACION, '//donativo [cantidad > 100]/nombre_p') AS
      PROYECTO
FROM DONACION_TAB
WHERE ID = 2;
```

9.5.2. Javier.

- El nombre de los proyectos que están financiados:

```
CREATE VIEW proy_financiados AS
    SELECT EXTRACT(DONACION,
      'donacion/proyecto[@estado="financiado"]/nombre_p/text()')
            AS FINANCIADO
    FROM DONACION_TAB
    WHERE ID = 1;
```

- El nombre de los inversores que tienen más saldo que Antonio García:

```
CREATE VIEW saldo_gt_Antonio_Garcia AS
    SELECT EXTRACT(DONACION, '//inversor[saldo>//inversor[nombre_i = "Antonio
       Garcia"]/saldo/text()]/nombre_i/text()') AS INVERSOR_GT
    FROM DONACION_TAB
    WHERE ID = 1;
```

9.6. Consultas XQuery.

9.6.1. Yasín.

- Consulta 1: Mostrar los proyectos cuya área es TIC.

```
CREATE VIEW PROYECTOS_TIC AS
SELECT XMLQUERY('for $i in //proyecto
                 where $i/area = "TIC"
                 return $i/nombre_p/text()'
                PASSING DONACION RETURNING CONTENT) AS PROYECTOS
FROM DONACION_TAB
WHERE ID = 2;
```

- Consulta 2: Suma de todas las donaciones

```
CREATE VIEW TOTAL_DONACIONES AS
SELECT XMLQUERY('for $i in //donacion
                 let $don := $i//cantidad
                 let $sum := sum($don)
                 return $sum'
                PASSING DONACION RETURNING CONTENT) AS TOTAL_DONACIONES
FROM DONACION_TAB
WHERE ID = 2;
```

9.6.2. Javier.

- Las áreas de los proyectos con rating "A":

```
CREATE VIEW RATING_AREAS AS
      SELECT XMLQUERY('for $i in //proyecto
                       where $i/rating = "A"
                       return $i/area'
              PASSING DONACION RETURNING CONTENT
      ) AS PROYECTOS_A
      FROM DONACION_TAB
      WHERE ID = 1;
```

- Sumar y mostrar (let) la cantidad total donada al proyecto "ReBur":

```
CREATE VIEW DONACION_TOTAL_ReBur AS
      SELECT XMLQUERY('for $i in /donacion
                       let $don := $i//cantidad[..[nombre_p="ReBur"]]
                       return <donacion_total>{sum($don)}</donacion_total>'
              PASSING DONACION RETURNING CONTENT
      ) AS DONACION_ReBur
      FROM DONACION_TAB
      WHERE ID = 1;
```

9.7. Actualización del fichero XML.

9.7.1. Yasín.

- Inserción: Insertar una donación

```
UPDATE DONACION_TAB
SET DONACION = APPENDCHILDXML(DONACION, '/donacion',
    xmltype(
        '<donativo>
           <nombre_p>ALVE</nombre_p>
           <dni>18194574J</dni>
           <cantidad>150</cantidad>
         </donativo>'
    )
)
WHERE ID = 2;
```

- Modificación: Modificar la cantidad donada a ALVE por 200€

```
UPDATE DONACION_TAB
SET DONACION =
UPDATEXML(DONACION,'/donacion/donativo[nombre_p="ALVE"][2]/cantidad/text()',
200)
WHERE ID = 2;
```

- Borrado: Borrar la donación a ALVE.

```
UPDATE DONACION_TAB
SET DONACION = DELETEXML(DONACION, '/donacion/donativo[nombre_p="ALVE"][2]')
WHERE ID = 2;
```

9.7.2. Javier.

- Inserción:

```
UPDATE DONACION_TAB
SET DONACION = APPENDCHILDXML(DONACION, '/donacion',
    xmltype(
        '<donativo>
            <nombre_p>ReBur</nombre_p>
            <dni>16161616F</dni>
                <cantidad>600</cantidad>
         </donativo>'
    )
)
WHERE ID = 1;
```

- Modificación:

```
UPDATE DONACION_TAB
SET DONACION = UPDATEXML(DONACION,
'/donacion/donativo[nombre_p="ReBur"][2]/cantidad/text()', 700)
WHERE ID = 1;
```

- Borrado:

```
UPDATE DONACION_TAB
SET DONACION = DELETEXML(DONACION, '/donacion/donativo[nombre_p="ReBur"][2]')
WHERE ID = 1;
```