



Base de Datos II [G4011229] [2020/2021]



[3 de mayo de 2021]

Aldrey Blanco, Óscar ; oscar.aldrey@rai.usc.es

Baleirón País, Diego; diego.baleiron@rai.usc.es

Baquero Domínguez, Alex ; alex.baquero@rai.usc.es

Cacheiro Pérez, Diego, diego.cacheiro@rai.usc.es

Carsi González, Ana ; ana.carsi@rai.usc.es

Eloi Corral López, eloi.corral@rai.usc.es

Cid Domínguez, Manuel, manuel.cid@rai.usc.es

Contenido

1. Introducción	4
1.2 Funcionalidades.....	4
2. Modelo de datos	7
2.1. Modelo Entidad Relación	7
2.2. Modelo Relacional.....	7
3. Script de la creación de la base de datos	10
3.1 Creación tabla de datos.....	10
3.1.1 Tabla Usuarios	10
3.1.2 Tabla Inversores	10
3.1.3 Empresas	11
3.1.4 Tabla Oferta Venta	11
3.1.5 Tabla Poseer	12
3.1.6 Tabla Compraventa	12
3.1.7 Tabla Pagos Dividendos	13
3.1.8 Anuncios	13
3.1.9 TablaActualizacionesPrecios	14
3.1.10 Monederos	14
3.1.11 CompraVentaCriptos	14
3.1.12 HacerOfertaVentaCripto	15
3.2 Inserción de datos	16
4. Interfaz	18
4.1 Ventana Autentificación	18
4.2 Registro Inversores.....	19
4.3 Registro Empresa.....	20
4.4 Mi Perfil Inversores	21
4.5 Mi Perfil Empresa	22
4.5.1 Mi Perfil	22
4.5.2 Mis Anuncios	23
4.6 Crear Anuncio.....	24
4.7 Ventana Pago Por Acción	24
4.8 Ventana Selección Pago Dividendos	25
4.9 Ventana Principal	26
4.9.1 Ventana Principal Inversor	26
4.9.2 Ventana Principal Empresa	27
4.10 Ventana Gestión Criptomonedas	28

1. Introducción

4.11 Ventana Vender Criptomonedas	29
4.12 Ventana Comprar Criptomonedas	30
4.13 Ventana Historial Oferta Venta	31
4.14 Ventana Generar Acciones	32
4.15 Ventana Comprar Acciones	33
4.16 Retirar Acciones.....	34
4.17 Ventana Aviso.....	34
4.18 Ventana Administrador	35
4.19 Ventana Añadir Fondos	36
4.20 Cambiar Contraseña	36
4.20 Ventana Realizar.....	37
5. Desarrollo de funcionalidades.....	38
6. Apéndice	76

1. Introducción

Un grupo de estudiantes de la ETSE, han sido contratados por Stonks para realizar una aplicación de comercio de acciones, donde inversores y empresas pueden comprar/vender acciones y vivir una experiencia inolvidable.

1.2 Funcionalidades

TIPOS DE USUARIOS

- **Regulador:** Usuario encargado de verificar el funcionamiento correcto del mercado de valores, especialmente en lo referido a la gestión de las cuentas de las empresas e inversores (modificaciones manuales de los saldos de las cuentas) y establecimiento de la comisión de compra/venta. No es necesario ningún dato de identificación personal ya que se trata de una persona jurídica. Accederá al sistema mediante identificación usuario/password.
- **Empresas:** Organizaciones que desean ofrecer participaciones sobre sus beneficios comerciales futuros. Se necesitan los siguientes datos de identificación: CIF, nombre comercial, dirección y teléfono de contacto. Podrá realizar acciones de compra/venta sobre las participaciones de las empresas autorizadas en el sistema (propias o ajenas). Accederá al sistema mediante identificación usuario/password.
- **Inversores:** Personas físicas que desean invertir mediante la negociación con participaciones de empresas. Se necesitan los siguientes datos de identificación: DNI, nombre y apellidos, dirección y teléfono de contacto. Podrá realizar acciones de compra/venta sobre las participaciones de las empresas autorizadas en el sistema. Accederá al sistema mediante identificación usuario/password.

FUNCIONALIDADES DE USUARIOS

- **[FU_01] Acceso Sistema:** El sistema solicitará a los usuarios su identificador y su clave de acceso. Si la información suministrada es correcta se accederá a la información de usuario y este podrá realizar las operaciones asociadas a su tipo de usuario. En caso contrario, se informará de un error en las credenciales.
- **[FU_02] Solicitud Registro:** Cualquier empresa/inversor podrá solicitar el alta en el mercado de valores mediante el suministro de sus datos de identificación. En este proceso de solicitud de registro, el usuario propondrá un identificador y una clave de acceso. En el caso de que el identificador ya exista en el sistema se informará al usuario y se volverá a solicitar. Si todos los datos son correctos se registrará la solicitud. Quedará pendiente de confirmar por parte del regulador.
- **[FU_03] Confirmación Registro:** Una solicitud de registro se convertirá en un registro efectivo cuando el regulador lo autorice. Una vez confirmado el registro, el usuario podrá acceder a una cuenta dónde se almacenarán sus fondos disponibles y a una cartera de participaciones dónde se almacenarán las participaciones que posee de cada empresa.
- **[FU_04] Modificación Datos:** Cualquier empresa/inversor podrá modificar sus datos de identificación en cualquier momento. También podrá modificar su identificador (siempre que siga siendo único en el sistema) y su clave de acceso. En el caso de que el identificador propuesto ya exista en el sistema se informará de un error en la modificación de datos.

1. Introducción

- **[FU_05] Modificación Saldo Cuenta:** Las modificaciones en el saldo de las cuentas de los usuarios, consecuencia de transferencias económicas recibidas y/o emitidas por la administración del mercado de valores, serán realizadas por el regulador tras la comprobación de la correspondiente transferencia (en la base de datos no se almacenará ninguna información sobre dichas transferencias). Los usuarios no podrán realizar modificaciones manuales en los saldos de sus cuentas (las modificaciones serán consecuencia de la realización de operaciones).
- **[FU_06] Solicitud Baja:** Cualquier empresa/inversor podrá solicitar su baja en el sistema en cualquier momento. Quedará pendiente de confirmar por parte del regulador.
- **[FU_07] Confirmación Baja:** La baja de un usuario debe ser confirmada por el regulador, una vez verifique que el usuario no posee participaciones y que el saldo de su cuenta está a cero. En el caso de que el usuario tenga participaciones se rechazará la baja. En el caso de que el saldo no sea cero se pondrá a cero (suponiendo la realización de la correspondiente transferencia) y se tramitará la baja.

FUNCIONALIDADES DE COMERCIO

- **[FC_01] Alta Participaciones:** Las empresas registradas podrán dar de alta el número de participaciones que deseen. Las participaciones dadas de alta por una empresa se incorporarán a su propia cartera de participaciones. Por simplicidad del sistema, las participaciones no se identificarán de forma unitaria. Sólo se registrará la cantidad total de participaciones, de cada empresa, que posee cada empresa/inversor y el número total de participaciones involucradas en cada operación de compra/venta.
- **[FC_02] Baja Participaciones:** Las empresas registradas podrán dar de baja el número de participaciones que deseen, siempre y cuando dispongan de ellas en su propia cartera. Las participaciones dadas de baja por una empresa se eliminarán del sistema disminuyendo el número total de participaciones existentes.
- **[FC_03] Comisión Compra/Venta:** El regulador del mercado fijará una comisión sobre las operaciones de compra/venta. El importe de la comisión de compra/venta deberá ser público y se informará al vendedor en el momento de realizar la oferta de venta. El importe de dicha comisión, a deducir del ingreso al vendedor, será ingresado en la cuenta especial del regulador. La comisión a aplicar a una operación de compra/venta será la disponible de forma pública en el sistema en el momento de poner en el mercado la oferta de venta.
- **[FC_04] Venta Participaciones:** Cualquier usuario, inversor y/o empresa, poseedor de participaciones de alguna empresa podrá ponerlas a la venta. El total de participaciones puestas a la venta por un usuario no puede ser superior al número de participaciones que posee dicho usuario. Para ello deberá indicar la empresa, el número global de participaciones y el precio de venta. Cuando las participaciones se vendan, recibirá en su cuenta el importe de la venta menos la comisión fijada por el mercado de valores. Las ofertas de venta permanecerán en el mercado de forma permanente salvo que sean retiradas por el usuario que las realice.
- **[FC_05] Baja Venta Participaciones:** El usuario que ha realizado una oferta de venta podrá eliminarla en cualquier momento. En el caso de ejecuciones parciales de la oferta de venta, la baja afectará a la parte de la oferta no ejecutada.
- **[FC_06] Compra Participaciones:** Cualquier usuario podrá realizar ofertas de compra, siempre y cuando disponga de saldo suficiente en su cuenta. Para ello deberá indicar la empresa, el número global de participaciones y el precio máximo de compra. La compra

se realizará de forma completa si hay suficientes participaciones a la venta al precio indicado o inferior (una compra puede completarse a partir de varias ofertas de venta; en ese caso, se seleccionarán las ofertas de venta por orden de precio y de antigüedad) o de forma parcial si hay menos participaciones a la venta, al precio fijado o inferior, de las indicadas en la orden de compra. Las órdenes de compra no quedarán en el mercado una vez cruzadas con las ofertas de venta existentes, tanto se hayan realizado de forma parcial o, incluso, no se hayan podido realizar por no existir oferta adecuada.

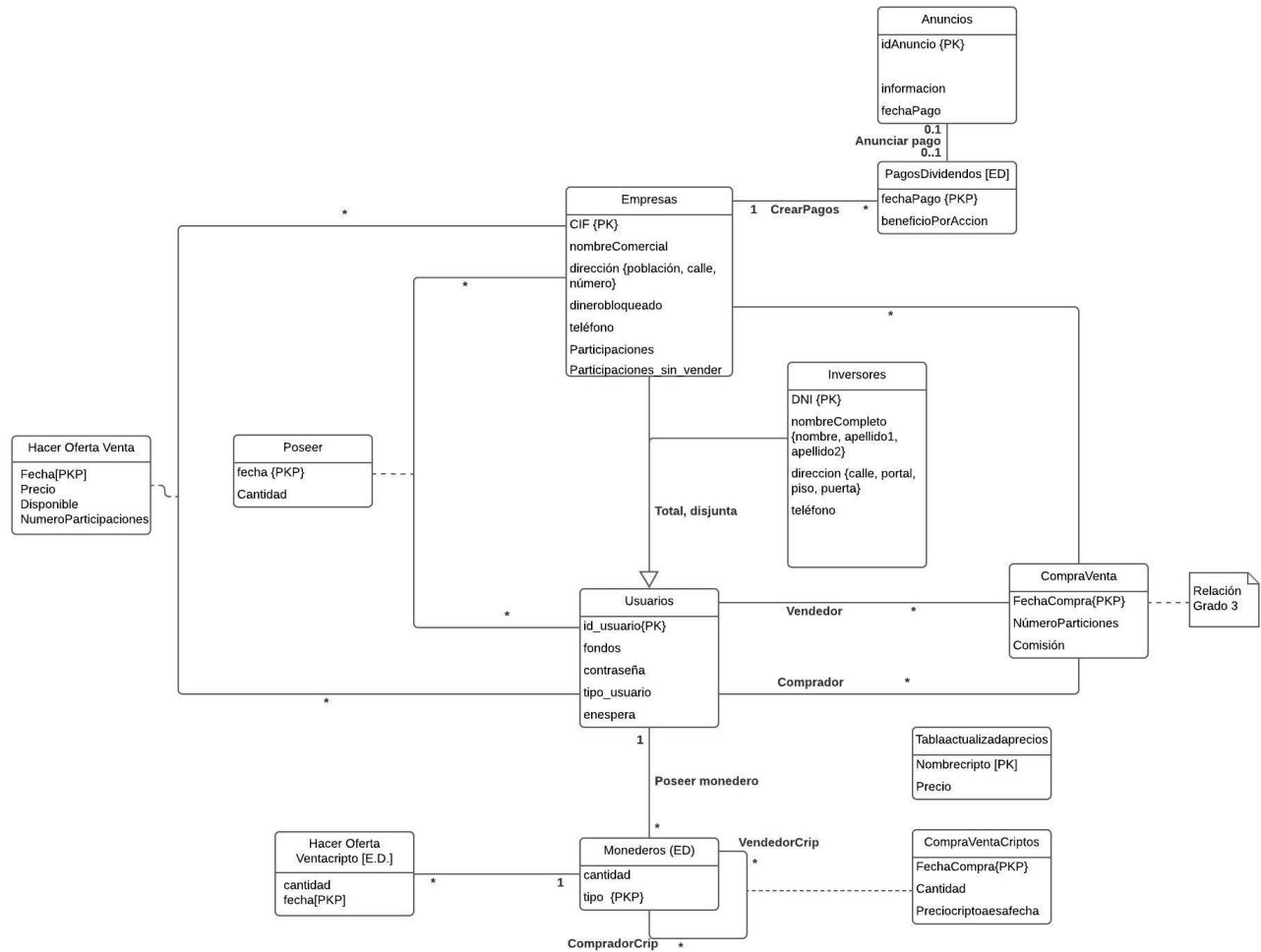
- **[FC_07] Alta Pago Beneficios:** Las empresas registradas podrán publicar información sobre sus pagos futuros de beneficios asociados a las participaciones que están en negociación en el mercado. Para ello deberá indicar la fecha de pago y el beneficio a abonar por cada participación (puede ser saldo en efectivo, participaciones o una combinación de ambas). Para completar el alta deberá tener en su cuenta el importe suficiente (en saldo en efectivo y/o en participaciones) para hacer frente al pago de beneficios a publicar. El importe de dicho pago se bloqueará (no estará disponible para operaciones) hasta la eliminación del anuncio de pago o hasta la realización del pago.
- **[FC_08] Baja Pago Beneficios:** La eliminación de información sobre pagos de beneficios sólo podrá ser realizada por el regulador del mercado de valores a petición de la empresa que lo ha realizado (en la base de datos no se almacenará ninguna información sobre dichas peticiones de baja).
- **[FC_09] Pago Beneficios:** Las empresas podrán realizar pagos de beneficios asociados a participaciones en cualquier momento, con anuncio previo o sin anuncio previo. En el caso de anuncio previo, en el momento de realizar el pago se eliminará el anuncio asociado y se realizará el pago con los fondos (saldo y/o participaciones) previamente bloqueados. En el caso de que no exista anuncio previo, el pago se realizará directamente, si la empresa dispone de fondos suficientes en su cuenta.

FUNCIONALIDADES EXTRA

- **[FEXTRA_01] Criptomonedas:** Compraventa de criptomonedas: Se le permite a cada usuario la compra-venta de criptomonedas, para ello el usuario puede tener de 0 a N monederos de criptomonedas y comprar criptomonedas a otros usuarios según el precio establecido en una tabla que recoge el precio actual de cada criptomoneda existente.
- **[FEXTERA_02] Encriptación contraseñas:** A través de una función has, se obtendrá una cadena de caracteres, por cada contraseña en la base de datos.
- **[FEXTRA_03] Lectura no comprometida:** Se permitirá leer el listado de solicitudes de cuenta de forma no comprometida, mejorando el rendimiento sin altos riesgos, ya que de no aceptarse la solicitud en ese trámite se hará en la siguiente bajo la premisa de que no se producirán errores al tratar de otorgar una cuenta a una inversor o empresa que ya la postean.

2. Modelo de datos

2.1. Modelo Entidad Relación



2.2. Modelo Relacional

USUARIOS (id_usuario, contraseña, fondos, tipo_usuario, enespera)

PK: id_usuario

INVERSORES (id_usuario, DNI, nombre, apellido1, apellido2, calle, portal, piso, puerta, teléfono)

PK: DNI, id_usuario

FK: id_usuario REFERENCIA Usuarios(id_usuario)

EMPRESAS(idusuario, CIF, nombreComercial, ciudad, calle, número, teléfono, participaciones, participacionessin vender, dinerobloqueado)

PK: CIF, idusuario

FK: idusuario REFERENCIA Usuarios(idusuario)

HACEROFERTAVENTA(fechaOferta, Precio, numparticipaciones, disponibilidad, Empresa, idEmpresa, idUsuario)

PK: fechaOferta, empresa, idempresa, idusuario

FK: empresa REFERENCIA Empresas(CIF)

FK: idempresa REFERENCIA Empresas(idusuario)

FK: idUsuario REFERENCIA Usuarios(id_usuario)

POSEER(Empresa, idEmpresa, idUsuario, Fecha, Cantidad)

PK: Empresa, idEmpresa, idUsuario, Fecha

FK: Empresa REFERENCIA Empresa(CIF)

FK: idEmpresa REFERENCIA Empresa(idusuario)

FK: idUsuario REFERENCIA Usuarios(id_usuario)

COMPRAVENTA(idUsuarioVendedor, id_UsuarioComprador, Empresa, idEmpresa, FechaCompra, NumeroParticipaciones, comisión)

PK: IdUsuarioVendedor, IdUsuarioComprador, FechaCompra, Empresa, idEmpresa

FK: Empresa REFERENCIA Empresas(CIF)

FK: idEmpresa REFERENCIA Empresas(idusuario)

FK: IdUsuarioComprador REFERENCIA Usuarios(id_usuario)

FK: IdUsuarioVendedor REFERENCIA Usuarios(id_usuario)

PAGOSDIVIDENDOS(Empresa, idEmpresa, FechaPago, beneficioPorAccion)

PK: FechaPago, Empresa, idEmpresa

FK: Empresa REFERENCIA Empresa(CIF)

FK: idEmpresa REFERENCIA Empresa(idusuario)

ANUNCIOS(idAnuncio, información, fechaPublicacion, fechaPago, Empresa)

2. Modelo de datos

PK: IdAnuncio

FK: fechaPublicacion REFERENCIA PagosDividendos(FechaPublicación)

FK: fechaPago REFERENCIA PagosDividendos(FechaPago)

FK: Empresa REFERENCIA PagosDividendos(Empresa)

FK: idEmpresa REFERENCIA PagosDividendos(idEmpresa)

TABLA ACTUALIZACIONES PRECIOS(nombrecripto, precio)

PK: nombrecripto

MONEDEROS (id_usuario, tipo, cantidad)

PK: tipo, id_usuario

FK: id_usuario REFERENCIA Usuarios(id_usuario)

COMPRAVENTA CRIPTOS (idComprador, idVendedor, tipo, fechaCompra, cantidad, precioCriptoEsaFecha)

PK: idComprador, idVendedor, fechaCompra, tipo

FK: idComprador REFERENCIA monederos (id_usuario)

FK: idVendedor REFERENCIA monederos(id_usuario)

FK: tipo REFERENCIA monederos(tipo) (Referencia al monedero del comprador)

HACER OFERTAVENTA CRIPTO(fechaOferta, cantidad, tipo, idusuario)

PK: fechaOferta, tipo, idusuario

FK: tipo REFERENCIA monederos(tipo)

FK: idusuario REFERENCIA monederos(id_usuario)

3. Script de la creación de la base de datos

3.1 Creación tabla de datos

El modelo de integridad referencial empleados han sido ON DELETE RESTRICT y ON UPDATE CASCADE, que se encargan de garantizar que, en caso de eliminar o actualizar una tupla, las tuplas referenciadas cambien también.

3.1.1 Tabla Usuarios

```
--Usuarios
CREATE TABLE public.usuarios
(
    id_usuario character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    contraseña character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    fondos real NOT NULL,
    tipo_usuario character varying(15) COLLATE pg_catalog."default"
NOT NULL,
    enespera integer NOT NULL,
    CONSTRAINT usuario_pkey PRIMARY KEY (id_usuario),
    CONSTRAINT usuario FOREIGN KEY (id_usuario)
        REFERENCES public.usuarios (id_usuario) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

3.1.2 Tabla Inversores

```
CREATE TABLE public.inversores
(
    idusuario character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    dni character varying(9) COLLATE pg_catalog."default" NOT NULL,
    nombre character varying(15) COLLATE pg_catalog."default" NOT
NULL,
    apellido1 character varying(15) COLLATE pg_catalog."default" NOT
NULL,
    apellido2 character varying(15) COLLATE pg_catalog."default" NOT
NULL,
    calle character varying(20) COLLATE pg_catalog."default" NOT NULL,
    portal integer NOT NULL,
    piso integer NOT NULL,
    puerta character varying(2) COLLATE pg_catalog."default" NOT NULL,
    telefono integer NOT NULL,
    CONSTRAINT inversores_pkey PRIMARY KEY (idusuario, dni),
    CONSTRAINT "idUsuario" FOREIGN KEY (idusuario)
        REFERENCES public.usuarios (id_usuario) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT
        NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

3. Script de la creación de la base de datos

3.1.3 Empresas

```
CREATE TABLE public.empresas
(
    idusuario character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    cif character varying(9) COLLATE pg_catalog."default" NOT NULL,
    nombrequercomercial character varying(20) COLLATE pg_catalog."default"
NOT NULL,
    ciudad character varying(15) COLLATE pg_catalog."default" NOT
NULL,
    calle character varying(50) COLLATE pg_catalog."default" NOT NULL,
    numero integer NOT NULL,
    telefono integer NOT NULL,
    participaciones integer NOT NULL,
    participacionessinvender integer NOT NULL,
    dinerobloqueado real NOT NULL,
    CONSTRAINT empresas_pkey PRIMARY KEY (idusuario, cif),
    CONSTRAINT idusuario FOREIGN KEY (idusuario)
REFERENCES public.usuarios (id_usuario) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

3.1.4 Tabla Oferta Venta

```
CREATE TABLE public.hacerofertaventa
(
    fechaoferta timestamp without time zone NOT NULL,
    precio real NOT NULL,
    numparticipaciones integer NOT NULL,
    disponibilidad boolean NOT NULL,
    empresa character varying(9) COLLATE pg_catalog."default" NOT
NULL,
    idempresa character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    idusuario character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    CONSTRAINT hacerofertaventa_pkey PRIMARY KEY (fechaoferta,
empresa, idempresa, idusuario),
    CONSTRAINT empresas FOREIGN KEY (empresa, idempresa)
REFERENCES public.empresas (cif, idusuario) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT,
    CONSTRAINT idusuario FOREIGN KEY (idusuario)
REFERENCES public.usuarios (id_usuario) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

3.1.5 Tabla Poseer

```
CREATE TABLE public.poseer
(
    empresa character varying(9) COLLATE pg_catalog."default" NOT
NULL,
    idempresa character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    idusuario character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    fecha timestamp without time zone NOT NULL,
    cantidad integer NOT NULL,
    CONSTRAINT poseer_pkey PRIMARY KEY (empresa, idempresa, idusuario,
fecha),
    CONSTRAINT empresa FOREIGN KEY (empresa, idempresa)
REFERENCES public.empresas (cif, idusuario) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT,
    CONSTRAINT idusuario FOREIGN KEY (idusuario)
REFERENCES public.usuarios (id_usuario) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

3.1.6 Tabla Compraventa

```
CREATE TABLE public.compraventa
(
    idusuariovendedor character varying(20) COLLATE
pg_catalog."default" NOT NULL,
    idusuariocomprador character varying(20) COLLATE
pg_catalog."default" NOT NULL,
    empresa character varying(9) COLLATE pg_catalog."default" NOT
NULL,
    idempresa character varying(20) COLLATE pg_catalog."default" NOT
NULL,
    fechacompra timestamp without time zone NOT NULL,
    numeroparticipaciones integer NOT NULL,
    comision real NOT NULL,
    CONSTRAINT compraventa_pkey PRIMARY KEY (idusuariovendedor,
idusuariocomprador, empresa, idempresa, fechacompra),
    CONSTRAINT empresa FOREIGN KEY (empresa, idempresa)
REFERENCES public.empresas (cif, idusuario) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT,
    CONSTRAINT idusuariocompr FOREIGN KEY (idusuariocomprador)
REFERENCES public.usuarios (id_usuario) MATCH SIMPLE
ON UPDATE CASCADE
ON DELETE RESTRICT,
    CONSTRAINT idusuariovend FOREIGN KEY (idusuariovendedor)
REFERENCES public.usuarios (id_usuario) MATCH SIMPLE
ON UPDATE CASCADE
)
```

3. Script de la creación de la base de datos

```
        ON DELETE RESTRICT
    )
    WITH (
        OIDS = FALSE
    )
    TABLESPACE pg_default;
```

3.1.7 Tabla Pagos Dividendos

```
CREATE TABLE public.pagosdividendos
(
    empresa character varying(9) COLLATE pg_catalog."default" NOT
    NULL,
    idempresa character varying(20) COLLATE pg_catalog."default" NOT
    NULL,
    fechapago timestamp without time zone NOT NULL,
    beneficio real NOT NULL,
    CONSTRAINT pagosdividendos_pkey PRIMARY KEY (empresa, idempresa,
    fechapago),
    CONSTRAINT empresa FOREIGN KEY (empresa, idempresa)
        REFERENCES public.empresas (cif, idusuario) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

3.1.8 Anuncios

```
CREATE TABLE public.anuncios
(
    idanuncio integer NOT NULL,
    informacion character varying(100) COLLATE pg_catalog."default"
    NOT NULL,
    fechapubli timestamp without time zone NOT NULL,
    fechapago timestamp without time zone NOT NULL,
    empresa character varying(9) COLLATE pg_catalog."default" NOT
    NULL,
    CONSTRAINT anuncios_pkey PRIMARY KEY (idanuncio)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

3.1.9 TablaActualizacionesPrecios

```
--TablaActualizacionesPrecios
CREATE TABLE public.tablaactualizacionesprecios
(
    nombrecripto character varying(20) COLLATE pg_catalog."default"
    NOT NULL,
    precio real NOT NULL,
    CONSTRAINT "Tablaactualizacionesprecios_pkey" PRIMARY KEY
    (nombrecripto)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

3.1.10 Monederos

```
CREATE TABLE public.monederos
(
    id_usuario character varying(20) COLLATE pg_catalog."default" NOT
    NULL,
    tipo character varying(20) COLLATE pg_catalog."default" NOT NULL,
    cantidad real NOT NULL,
    CONSTRAINT monederos_pkey PRIMARY KEY (id_usuario, tipo),
    CONSTRAINT id_usuario FOREIGN KEY (id_usuario)
    REFERENCES public.usuarios (id_usuario) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE RESTRICT
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

3.1.11 CompraVentaCiertos

```
CREATE TABLE public.compraventaciertos
(
    idcomprador character varying(20) COLLATE pg_catalog."default" NOT
    NULL,
    idvendedor character varying(20) COLLATE pg_catalog."default" NOT
    NULL,
    tipo character varying(20) COLLATE pg_catalog."default" NOT NULL,
    fechacompra timestamp without time zone NOT NULL,
    cantidad real NOT NULL,
    precio real NOT NULL,
    CONSTRAINT compraventaciertos_pkey PRIMARY KEY (idcomprador,
    idvendedor, tipo, fechacompra),
    CONSTRAINT "idComprador" FOREIGN KEY (idcomprador, tipo)
    REFERENCES public.monederos (id_usuario, tipo) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE RESTRICT,
    CONSTRAINT "idVendedor" FOREIGN KEY (idcomprador, tipo)
    REFERENCES public.monederos (id_usuario, tipo) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE RESTRICT
    NOT VALID
)
WITH (
    OIDS = FALSE
)
```

3. Script de la creación de la base de datos

```
TABLESPACE pg_default;
```

3.1.12 HacerOfertaVentaCripto

```
CREATE TABLE public.hacerofertaventacripto  
(  
    fechaoferta timestamp without time zone NOT NULL,  
    cantidad real NOT NULL,  
    tipo character varying(20) COLLATE pg_catalog."default" NOT NULL,  
    idusuario character varying(20) COLLATE pg_catalog."default" NOT  
NULL,  
    CONSTRAINT hacerofertaventacripto_pkey PRIMARY KEY (fechaoferta,  
idusuario, tipo),  
    CONSTRAINT usuario FOREIGN KEY (idusuario, tipo)  
        REFERENCES public.monederos (id_usuario, tipo) MATCH SIMPLE  
        ON UPDATE CASCADE  
        ON DELETE RESTRICT  
)  
WITH (  
    OIDS = FALSE  
)  
TABLESPACE pg_default;
```

3.2 Inserción de datos

```
-- USUARIOS
--contrasenas sin encriptar comentadas, necesarias para iniciar sesión
insert into usuarios values ('Inditex','326212', 90000, 'empresas',
0); --1234
insert into usuarios values ('Repsol','326212', 10000, 'empresas', 0);
--1234
insert into usuarios values ('Citroen','a3f38506', 80000, 'empresas',
0); --555a
insert into usuarios values ('a','a3f97', 80000, 'empresas', 1); --a
insert into usuarios values ('Diego','326212', 1000000, 'inversores',
0); --1234
insert into usuarios values ('Alex','a3f25186', 1000, 'inversores',
0); --1111
insert into usuarios values ('Oscar','a3f25700', 100, 'inversores',
1); --2222
insert into usuarios values ('b','a3f98', 100000, 'inversores', 1); --
b
insert into usuarios values ('0','a3f48',1000000, 'regulador', 0);--0
insert into usuarios values ('anaca', 'a3f53795', 10000, 'inversores',
0); --anaca

-- EMPRESAS
insert into empresas values ('Inditex', '44378195K', 'Inditex',
'Madrid', 'Neptuno', 1, 687450436, 0, 3000, 0);
insert into empresas values ('Repsol','54358695J','Repsol',
'Barcelona', 'Ramblas', 2, 676502348, 0, 2500, 0);
insert into empresas values ('Citroen','65781243H','Citroen',
'Vigo', 'Balaidos', 3, 987912412, 10, 1000, 0);
insert into empresas values ('a','98456321G','a', 'milladoiro', 'Rep
Argentina', 4, 536789654, 0, 4000, 0);

-- INVERSORES
insert into inversores values ('Diego','53194091J','Diego',
'Cacheiro', 'Perez', 'Neptuno', 124, 6, 'D', 674680404);
insert into inversores values ('Alex','54358695J','Alex', 'Baquero',
'Dominguez', 'Ramblas', 23, 5, 'G', 687549800);
insert into inversores values ('Oscar','65781243H','Oscar', 'Aldrey',
'Blanco','Balaidos', 33, 4, 'B', 677254687);
insert into inversores values ('b','98456321G','b', 'b',
'b','milladoiro', 45, 1, 'A', 536789654);

--hacerOfertaVenta
insert into hacerofertaventa values ('2021/04/19 12:00:00','50',100,
true, '44378195K', 'Inditex', 'Oscar');
insert into hacerofertaventa values ('2021/04/19 00:00:00','50',100,
false, '65781243H', 'Citroen', 'Diego');
insert into hacerofertaventa values ('2021/04/19 13:00:00','50',100,
true, '54358695J', 'Repsol', 'Alex');

--Poseer
insert into poseer values ('44378195K', 'Inditex', 'Oscar',
'2021/04/07 14:00:00', 100);
insert into poseer values ('54358695J', 'Repsol', 'Alex', '2021/04/10
20:00:00', 100);
insert into poseer values ('65781243H', 'Citroen', 'Diego',
'2021/04/13 04:00:00', 100);

--Compraventa
```


3. Script de la creación de la base de datos

```
insert into compraventa values ( 'Inditex', 'Oscar', '44378195K',
'Inditex', '2021/04/07 14:00:00', 100, 500);
insert into compraventa values ( 'Repsol', 'Alex', '54358695J',
'Repsol', '2021/04/10 20:00:00', 100, 500);
insert into compraventa values ( 'Citroen', 'Diego', '65781243H',
'Citroen', '2021/04/13 04:00:00', 100, 500);

--PagosDividendo
insert into pagosdividendos values ( '44378195K', 'Inditex',
'2021/04/07 14:00:00', 1000);
insert into pagosdividendos values ( '54358695J', 'Repsol',
'2021/04/10 20:00:00', 1000);
insert into pagosdividendos values ( '65781243H', 'Citroen',
'2021/04/13 04:00:00', 1000);

--Anuncios
insert into anuncios values (1,'Pago de Oscar a Inditex', '2021/04/19
12:00:00', '2021/04/07 14:00:00', '44378195K');
insert into anuncios values (2, 'Pago de Alex a Repsol', '2021/04/19
13:00:00', '2021/04/10 20:00:00', '54358695J');
insert into anuncios values (3, 'Pago de Diego a Citroen', '2021/04/19
00:00:00', '2021/04/13 04:00:00', '65781243H');

--TablaActualizacionesPrecios
insert into tablaactualizacionesprecios values ('Bitcoin',40000);
insert into tablaactualizacionesprecios values ('Ethereum',2000);
insert into tablaactualizacionesprecios values ('Dogecoin',10000);

--Monederos
insert into monederos values ('Diego','Bitcoin', 30);
insert into monederos values ('Alex','Bitcoin', 10);
insert into monederos values ('Alex','Ethereum', 10);
insert into monederos values ('Oscar','Ethereum', 20);
insert into monederos values ('Diego','Dogecoin', 10);

--compraventacriptos
insert into compraventacriptos values ('Diego','Alex','Bitcoin',
'2021/04/21 12:00:00',3,40000);
insert into compraventacriptos values ('Alex','Oscar','Ethereum',
'2021/04/15 16:00:00',1,2000);

--HacerOfertaVentaCripto
insert into hacerofertaventacripto values ('2021/04/14
20:00:00',1,'Ethereum', 'Oscar');
insert into hacerofertaventacripto values ('2021/04/14
20:00:00',3,'Bitcoin', 'Alex');
```

4. Interfaz

4.1 Ventana Autenticación

STONKS

DONDE TU DINERO
CRECE

ID

1

Contraseña

2

3 Completa todos los campos

4 REGISTRARSE 5 INICIAR SESIÓN

6 [Regístrame como empresa](#)

Esta es la primera ventana que se le muestra al usuario.

1. Se introduce la ID del usuario.
2. Se introduce la contraseña del usuario
3. En caso de algún error, aparece la etiqueta indicándoselo al usuario.
4. Permite al usuario registrarse como inversor.
5. Este botón comprueba que las credenciales son correctas, y da paso a la ventana Principal, la cuál será distinta en función del tipo de usuario que acceda (Inversor, Empresa o Regulador). El regulador es el usuario con ID "0" y Contraseña "0".
6. Permite al usuario registrarse como empresa.

4.2 Registro Inversores

The screenshot shows a web form titled "REGISTRO DE INVERSORES". It is divided into three main sections: "Datos de Cuenta", "Información Usuario", and "Dirección".

- Datos de Cuenta:** Contains two input fields: "ID:" (callout 1) and "CONTRASEÑA:" (callout 2).
- Información Usuario:** Contains five input fields: "NOMBRE:" (callout 3), "APELLIDO 1:" (callout 4), "APELLIDO 2:" (callout 5), "DNI:" (callout 6), and "TELÉFONO:" (callout 7).
- Dirección:** Contains three input fields: "CALLE:" (callout 8), "PORTAL:" (callout 9), and "PISO:" (callout 10). There is also a "PUERTA:" field (callout 11) which is partially visible.

At the bottom left, there is a red error message (callout 12): "El campo Teléfono no ha sido completado".

At the bottom right, there are three buttons: "REGISTRARSE" (callout 13), "LIMPIAR CAMPOS" (callout 14), and "VOLVER" (callout 15).

Se accede a esta ventana a través de la Ventana de Autenticación y nos permite registrar a un usuario como inversor.

1. Se introduce el ID del usuario
2. Se introduce la contraseña del usuario
3. Se introduce el nombre del usuario
4. Se introduce el 1º apellido del usuario
5. Se introduce el 2º apellido del usuario
6. Se introduce el DNI del usuario
7. Se introduce el teléfono del usuario
8. Se introduce la calle donde reside el usuario
9. Se introduce el portal donde reside el usuario.
10. Se introduce el piso donde reside el usuario.
11. Se introduce la puerta donde reside el usuario.
12. Etiqueta de error, en caso de algún campo vacío o no válido avisa al usuario.
13. Permite al usuario registrarse en la empresa si tiene todos los campos debidamente cubiertos.
14. Vacía todos los campos.
15. Cierra la ventana de Registro de Inversores

4.3 Registro Empresa

The screenshot shows a web form titled "Registro Empresa" with a close button (X) in the top right corner. The form is divided into two sections: "Datos de Cuenta" and "Información Usuario".

Datos de Cuenta

- 1. NOMBRE USUARIO: (text input field)
- 2. CONTRASEÑA: (password input field)

Información Usuario

- 3. NOMBRE EMPRESA: (text input field)
- 4. CIF: (text input field)
- 5. POBLACIÓN: (text input field)
- 6. CALLE: (text input field)
- 7. NÚMERO: (text input field)
- 8. TELÉFONO: (text input field)

At the bottom of the form are three buttons:

- 9. REGISTRARSE (orange button)
- 10. LIMPIAR CAMPOS (orange button)
- 11. VOLVER (orange button)

Se accede a esta ventana a través de la Ventana de Autenticación y nos permite registrar a una empresa en la aplicación.

1. Se introduce el nombre del usuario que usará la empresa.
2. Se introduce la contraseña del usuario que usará la empresa.
3. Se introduce el nombre de la empresa.
4. Se introduce el CIF de la empresa.
5. Se introduce la población a la que pertenece de la empresa.
6. Se introduce la calle en la que está ubicada la empresa.
7. Se introduce el número de la dirección de la empresa (debe de ser un número).
8. Se introduce el teléfono de la empresa (debe de ser un número).
9. Se registra a una empresa una vez completados todos los campos
10. Limpia los campos.
11. Cierra la ventana de Registro de Inversores

4.4 Mi Perfil Inversores

MI PERFIL

NOMBRE 1

APELLIDO 1 2 APELLIDO 2 3

ID 4 DNI 5

CALLE 6

PORTAL 7 PL 8 PUERTA 9

TELÉFONO 10

11 **GUARDAR** 12 **CAMBIAR CONTRASEÑA** 13 **DAR DE BAJA** 14 **VOLVER**

Se accede a esta ventana a través de la Ventana principal y nos permite ver el perfil de un inversor en la aplicación.

1. Se muestra el nombre del usuario que usará el inversor.
2. Se muestra el primer apellido del usuario que usará el inversor.
3. Se muestra el segundo apellido del usuario que usará el inversor.
4. Se muestra el id de la empresa.
5. Se muestra la población a la que pertenece del inversor.
6. Se muestra la calle del domicilio del inversor.
7. Se muestra el número de la dirección de la empresa (debe de ser un número).
8. Se muestra el piso del inversor (debe de ser un número).
9. Se muestra la puerta del piso en el que vive el inversor (opcional).
10. Se muestra el valor del teléfono del inversor (cadena numérica).
11. Guarda los datos modificados.
12. Botón que inicia la ventana de cambiar contraseñas
13. Botón que permite dar de baja a un usuario. Vuelve a la ventana de inicio sesión y envía una solicitud al administrador para dar de baja al usuario.
14. Botón que permite acceder a una nueva ventana en la que se puede cambiar la contraseña.
15. Botón que comprueba si el inversor tiene participaciones y en caso de que no las tenga le da de baja.
16. Permite volver a la pestaña principal de los inversores.

4.5 Mi Perfil Empresa

4.5.1 Mi Perfil

Se accede a través de la ventana principal de empresas y permite modificar los datos de la empresa.

1. Pestaña que muestra los datos de los inversores.
2. Pestaña que permite desarrollar y visualizar anuncios y pagos.
3. Área de texto que permite editar el nombre comercial de la empresa.
4. Área de texto que permite editar el id de la cuenta de la empresa.
5. Área de texto que permite editar el CIF de la empresa.
6. Área de texto que permite editar el número de teléfono de la empresa (tiene que ser una cadena de números).
7. Área de texto que permite modificar la población en la que está ubicada la empresa.
8. Área de texto que permite modificar la calle en la que está ubicada la empresa.
9. Área de texto que permite modificar el portal de la empresa.
10. Área de texto que muestra la cantidad de acciones que tiene la empresa en propiedad sin intención de vender.
11. Área de texto que muestra las acciones que tiene la empresa ofertada.
12. Botón que permite guardar todas las modificaciones introducidas en las áreas de texto.
13. Botón que permite acceder a una nueva ventana en la que se puede cambiar la contraseña.
14. Botón que comprueba si la empresa tiene participaciones y en caso de que no las tenga la da de baja.
15. Botón que permite volver a pestaña principal de empresas.

4. Interfaz

4.5.2 Mis Anuncios

The screenshot shows a web application window titled 'Mis Anuncios'. At the top, there are two tabs: 'Datos Perfil' (1) and 'Mis Anuncios' (2). Below the tabs, there is a search bar labeled 'Filtro:' (3) with a magnifying glass icon (4). Below the search bar is a table (5) with four columns: 'ID Anuncio', 'Descripción', 'Fecha Publicación', and 'Fecha Pago'. The table body is currently empty. At the bottom of the window, there are four orange buttons: 'BORRAR ANUNCIO' (6), 'REALIZAR PAGO' (7), 'CREAR ANUNCIO' (8), and 'VOLVER' (9).

Se accede a través de la ventana principal de empresas y permite realizar todas las operaciones relacionadas con los anuncios y los pagos.

1. Pestaña que muestra los datos de los inversores.
2. Pestaña que permite desarrollar y visualizar anuncios y pagos.
3. Área de texto que permite escribir cadenas de texto con el objetivo de usarlas luego para filtrar los anuncios.
4. Área de texto que permite usar el texto anterior para filtrar los anuncios por su nombre.
5. Tabla que permite ver diversos campos de los anuncios que emite la empresa como el id, una descripción, su fecha de publicación o su fecha de pago.
6. Elimina el anuncio.
7. Permite a la empresa realizar el pago del anuncio creado en la tabla. Para que funcione tiene que haber una fila seleccionada.
8. Permite a la empresa crear un anuncio para realizar pagos.
9. Botón que permite volver a la ventana principal de empresas.

4.6 Crear Anuncio

Se accede a través de la pestaña de mis anuncios (propia de las empresas) y permite crear un anuncio desde 0.

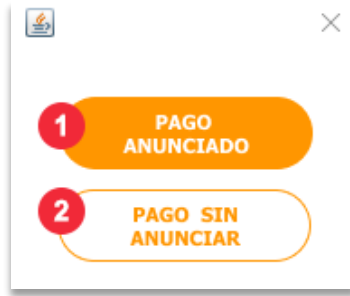
1. Área de texto que permite introducir una descripción del anuncio.
2. Menú que permite introducir la fecha en la que se realizará el pago en formato years-months-days.
3. Menú que permite guardar la hora a la que se realizará el pago.
4. Botón que permite salir de ventana sin hacer el anuncio.
5. Botón que permite crear un nuevo anuncio con todos los datos introducidos.

4.7 Ventana Pago Por Acción

Ventana que permite gestionar cuanto va a pagar la empresa por acción a cada accionista.

1. Área de texto que permite introducir la cantidad que pagará la empresa por acción a los inversores.
2. Botón que permite guardar el dato anterior.
3. Botón que permite salir de ventana sin fijar el precio del pago.

4.8 Ventana Selección Pago Dividendos



Ventana que permite crear pagos.

4. Botón que redirige al usuario a la ventana de gestión de anuncios 4.5.2 para poder crear uno o asociarle un pago a uno ya creado.
5. Botón que permite realizar un pago sin anuncio previo redirigiendo al usuario a la ventana 4.7

4.9 Ventana Principal

4.9.1 Ventana Principal Inversor

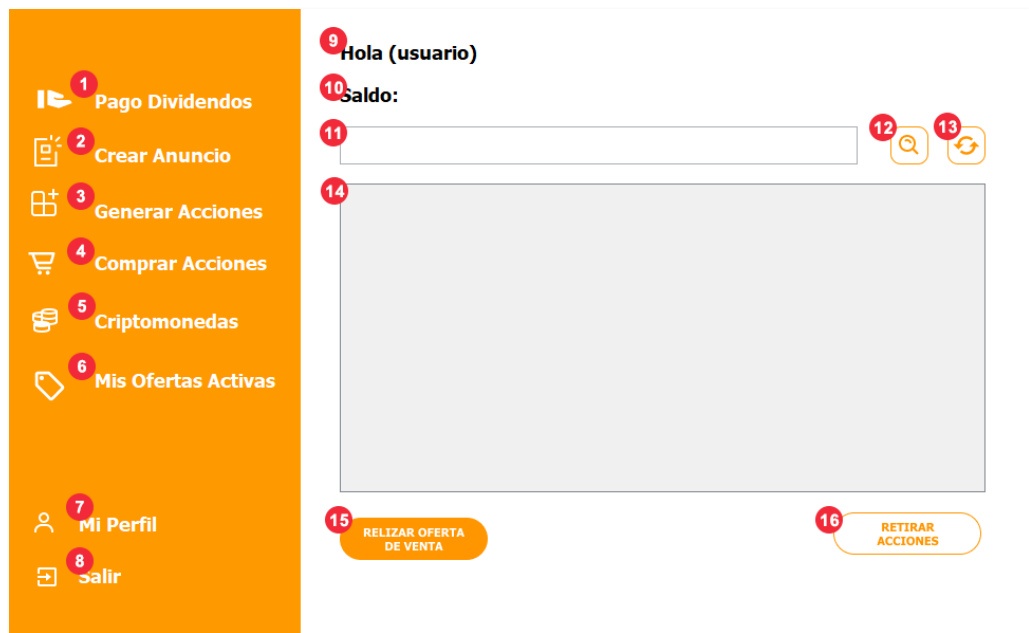


Ventana principal de los inversores desde la que se puede acceder a la mayoría de funcionalidades importantes de este tipo de usuario, se accede a través de la ventana de autenticación.

1. Botón que permite comprar acciones a través de una ventana en la que se mostrarán todas las ofertas existentes. [Ventana 4.14].
2. Botón para acceder a la compra-venta de criptomonedas. [Ventana 4.12]
3. Botón que permite acceder a una ventana en la que ver todas las ofertas de venta activas que tiene el usuario. [Ventana 4.14].
4. Botón que permite acceder a una zona en la que se pueden visualizar y modificar los datos del inversor. [Ventana 4.5]
5. Botón para cerrar la sesión. [Ventana 4.1]
6. Mensaje de bienvenida al inversor.
7. Mensaje donde aparece el saldo de la empresa.
8. Área de texto que permite escribir un texto para filtrar las filas de la tabla por el campo empresa.
9. Botón que permite usar el texto anterior para filtrar la tabla.
10. Botón que refresca la tabla imprimiendo todos los elementos de esta.
11. Tabla en la que se muestra información como el nombre de la empresa, la fecha de adquisición y la cantidad de las acciones que tiene el inversor.
12. Botón que permite realizar una oferta de venta de las acciones seleccionadas en la tabla. [Ventana 4.20].

4. Interfaz

4.9.2 Ventana Principal Empresa



En esta sección se muestra la ventana principal para un usuario registrado como inversor. Permite acceder a la mayoría de funcionalidades de este tipo de usuarios.

1. Botón que accede al pago de dividendos [Ventana 4.8].
2. Botón que crea un anuncio [Ventana 4.6].
3. Botón para generar acciones [Ventana 4.14].
4. Botón que permite comprar acciones a través de una ventana en la que se mostrarán todas las ofertas existentes.[Ventana 4.15]
5. Botón para acceder a la compra-venta de criptomonedas. [Ventana 4.12]
6. Botón para visualizar las ofertas activas.[Ventana 4.13]
7. Botón que permite acceder a una zona en la que se pueden visualizar y modificar los datos del inversor. [Ventana 4.5]
8. Botón para cerrar la sesión.[Ventana 4.1]
9. Mensaje de bienvenida a la empresa.
10. Mensaje donde aparece el saldo de la empresa.
11. Área de texto que permite escribir un texto para filtrar las filas de la tabla por el campo empresa.
12. Botón que permite usar el texto anterior para filtrar la tabla.
13. Botón que refresca la tabla imprimiendo todos los elementos de esta.
14. Tabla en la que se muestra información como el nombre de la empresa, la fecha de adquisición y la cantidad de las acciones que tiene el inversor.
15. Botón para ofertar acciones [Ventana 4.20].
16. Botón para retirar acciones de la propia empresa. [Ventana 4.16]

4.10 Ventana Gestión Criptomonedas



CRIPATOMONEDAS

1. Search bar

2. Search icon

3. Refresh icon

CriptoMoneda	Cantidad	Precio
Bitcoin	30.0	40000.0
Dogecoin	10.0	10000.0

4. Table showing cryptocurrency holdings

5. VENDER CRIPTOMONEDA button

6. COMPRAR CRIPTOMONEDA button

7. VOLVER button

Se accede a través de la ventana principal y permite acceder a la compra y venta de criptomonedas.

1. Filtrado de la tabla de la tabla monedero de un usuario.
2. Actualiza la tabla según el filtrado.
3. Refresca la tabla.
4. Tabla que muestra las diferentes criptomonedas que tiene un usuario.
5. Botón que accede a la oferta de criptomonedas.
15. Botón que accede a la compra de criptomonedas.
16. Botón que regresa a la ventana principal.

4.11 Ventana Vender Criptomonedas

¿Cuántas criptomonedas quieres vender?

1

MÁXIMO DE CRIPTOMONEDAS QUE PUEDE VENDER

2

3 **ACEPTAR** 4 **CANCELAR**

Se accede a través de la ventana principal de criptomonedas, desde el botón venta criptomonedas. Permite ofrecer a los usuarios una cantidad de criptomonedas.

1. Cantidad de criptomonedas que se quieren poner a la venta. No se puede ofertar más criptomonedas que su número máximo.
2. Número máximo de criptomonedas que se pueden vender.
3. Pone en oferta la cantidad de criptomonedas establecidas en [1] y vuelve a la ventana principal de criptomonedas
4. Cancela la oferta y vuelve a la ventana principal de criptomonedas.

4.12 Ventana Comprar Criptomonedas

Criptos Ofertadas

Filtro:

Nombre Usuario	Nombre Criptos	Cantidad Ofertada	Precio actual por c...	Fecha Publicación
Oscar	Ethereum	1.0	2000.0	2021-04-14 20:00:...
Alex	Bitcoin	3.0	40000.0	2021-04-14 20:00:...

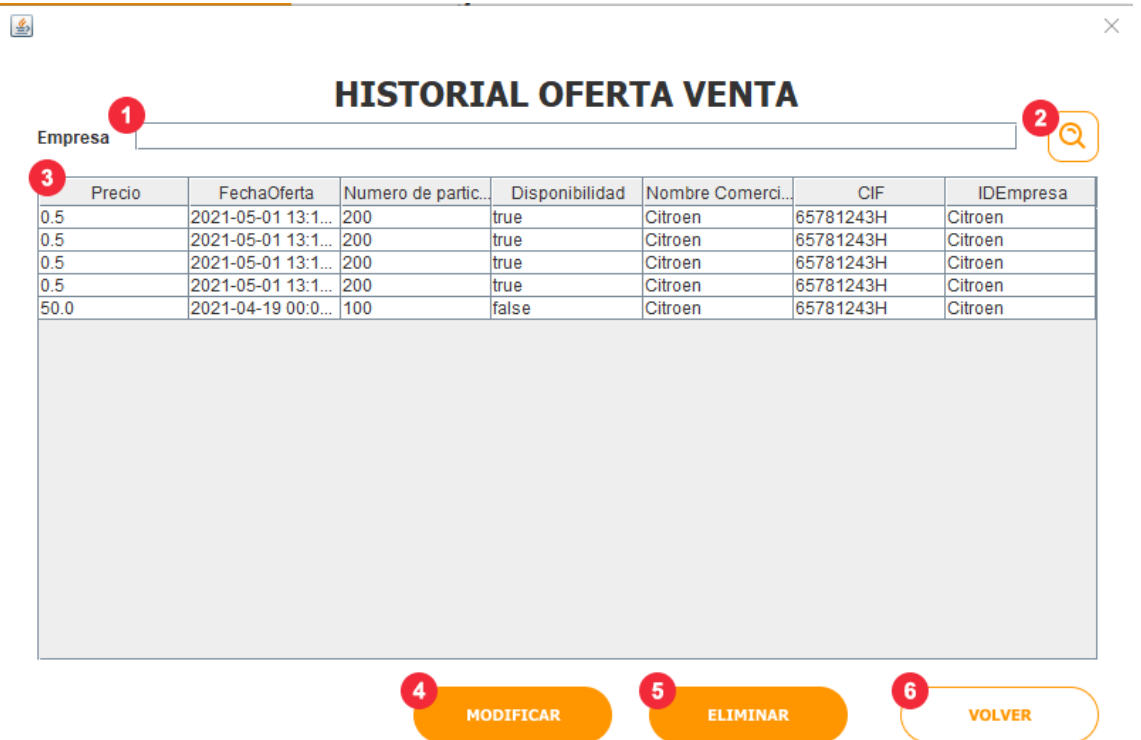
Número de criptos a comprar:

COMPRAR **SALIR**

Se accede a través de la ventana principal de criptomonedas, desde el botón compra criptomonedas. Permite comprar criptomonedas a otros usuarios.

1. Filtrado por tipo criptomoneda de la tabla ofertas.
2. Busca en la tabla ofertas según el filtro.
3. Refresca la tabla ofertas.
4. Tabla que muestra las ofertas de criptomonedas de los diferentes usuarios.
5. Cantidad que se desea comprar. Tiene que ser inferior o igual a la cantidad ofertada.
6. Confirma la compra.
7. Vuelve a la ventana principal de criptomonedas.

4.13 Ventana Historial Oferta Venta



HISTORIAL OFERTA VENTA

Empresa

Precio	FechaOferta	Numero de partic...	Disponibilidad	Nombre Comerci...	CIF	IDEmpresa
0.5	2021-05-01 13:1...	200	true	Citroen	65781243H	Citroen
0.5	2021-05-01 13:1...	200	true	Citroen	65781243H	Citroen
0.5	2021-05-01 13:1...	200	true	Citroen	65781243H	Citroen
0.5	2021-05-01 13:1...	200	true	Citroen	65781243H	Citroen
50.0	2021-04-19 00:0...	100	false	Citroen	65781243H	Citroen

Historial de las ofertas de venta activas de inversores o empresas.

1. Área de texto que permite escribir un texto para filtrar las ofertas de venta por el id de las empresas
2. Botón para usar el texto anterior para filtrar la tabla.
3. Tabla que muestra todas las ofertas activas del usuario que accede a la ventana.
4. Botón que permite modificar los datos de una oferta de venta.
5. Botón que permite eliminar la oferta de venta seleccionada en la tabla.
6. Botón que permite volver a la pestaña principal.

4.14 Ventana Generar Acciones

Se accede a través de la ventana principal, desde el botón Generar Acciones. Permite crear las acciones que se introducen.

1. Texto para introducir la cantidad de acciones
2. Cierra la ventana Generar Acciones y vuelve a la ventana principal

3. Llama al método Generar Acciones y actualiza la tabla de la ventana principal, en la cual aparecen las acciones. Por último, cierra la ventana y vuelve a la ventana principal

4.15 Ventana Comprar Acciones

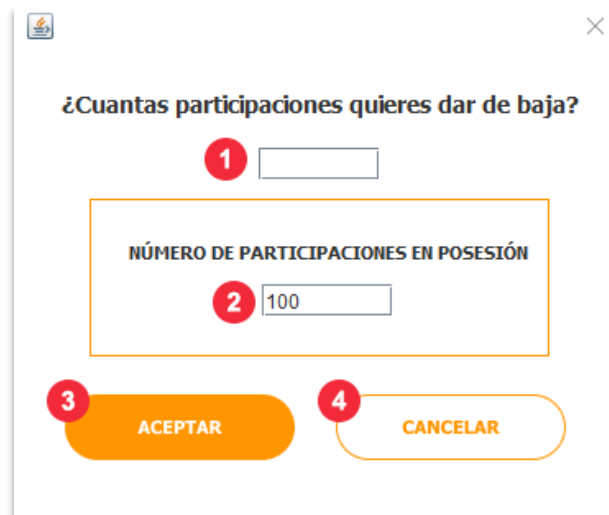
The screenshot shows a window titled "Acciones Ofertadas". At the top, there is a "Filtro:" label followed by a text input field (1). To the right of the input field are two buttons: a magnifying glass icon (2) and a circular arrow icon (3). Below the filter is a table (4) with three columns: "IdEmpresa", "CIF Empresa", and "Cantidad Acciones". The table contains two rows: "Inditex" with CIF "44378195K" and quantity "5", and "Repsol" with CIF "54358695J" and quantity "100". Below the table is a large empty rectangular area. At the bottom, there is a label "Número de acciones a comprar:" (5) followed by a text input field. To the right of the input field are two buttons: a yellow "COMPRAR" button (6) and a yellow "SALIR" button (7).

IdEmpresa	CIF Empresa	Cantidad Acciones
Inditex	44378195K	5
Repsol	54358695J	100

Se accede a través de la ventana principal, pulsando el botón comprar acciones.

1. Texto para filtrar en la tabla por nombre.
2. Botón de búsqueda, actualiza la tabla usando el texto escrito en la barra (1).
3. Botón que actualiza la tabla.
4. Tabla que muestra las acciones que están a la venta en este momento.
5. Cuadro para indicar cuantas acciones se desean comprar.
6. Se encarga de realizar la compra, teniendo en cuenta el saldo del usuario y disponibilidad.
7. Cierra la ventana.

4.16 Retirar Acciones

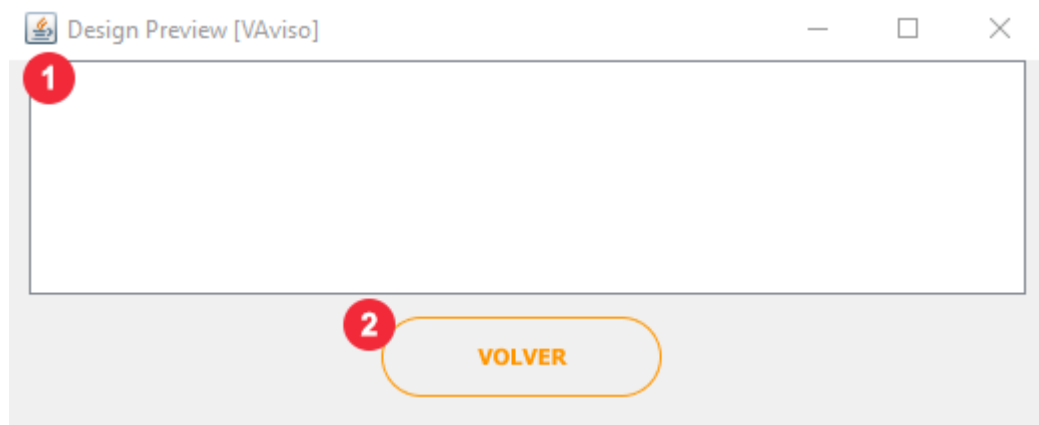


A modal window titled "¿Cuántas participaciones quieres dar de baja?" (How many shares do you want to withdraw?). It contains a text input field labeled with a red circle '1'. Below it, a box labeled "NÚMERO DE PARTICIPACIONES EN POSESIÓN" (Number of shares in possession) contains a text input field with the value '100', labeled with a red circle '2'. At the bottom, there are two orange buttons: "ACEPTAR" (Accept) labeled with a red circle '3' and "CANCELAR" (Cancel) labeled with a red circle '4'.

Se accede a través de la ventana principal, desde el botón retirar acciones. Permite retirar las ofertas de una misma empresa.

1. Cantidad de acciones que se quieren poner a la venta. No se pueden retirar más acciones que su número máximo.
2. Número máximo de acciones que se pueden vender.
3. Retira la cantidad de acciones establecidas en [1] y vuelve a la ventana principal.
4. Cancela el retiro de acciones y vuelve a la ventana principal.

4.17 Ventana Aviso

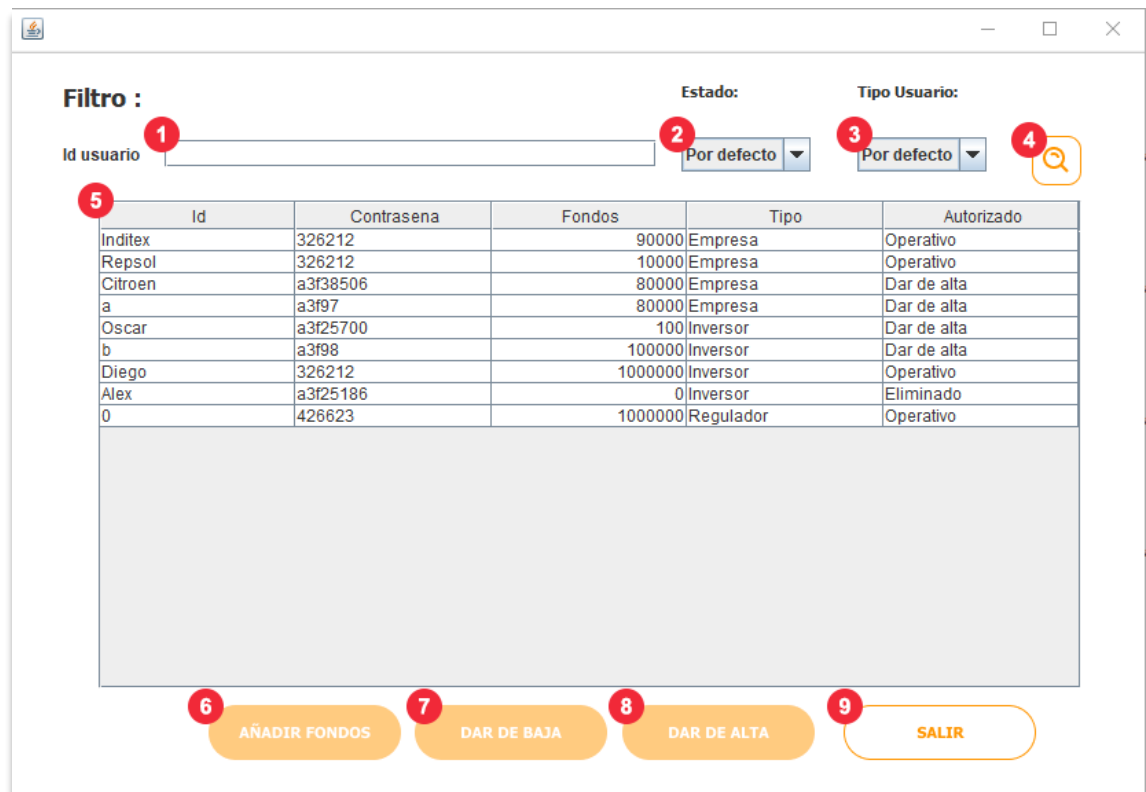


A design preview window titled "Design Preview [VAviso]". It features a large empty rectangular area labeled with a red circle '1'. At the bottom center, there is an orange button labeled "VOLVER" (Return) labeled with a red circle '2'.

En esta interfaz se muestran las diferentes excepciones y avisos que se producen en la aplicación.

1. Muestra el mensaje.
2. Vuelve a la aplicación.

4.18 Ventana Administrador



Filtro :

Id usuario

Estado: Por defecto ▼

Tipo Usuario: Por defecto ▼

5

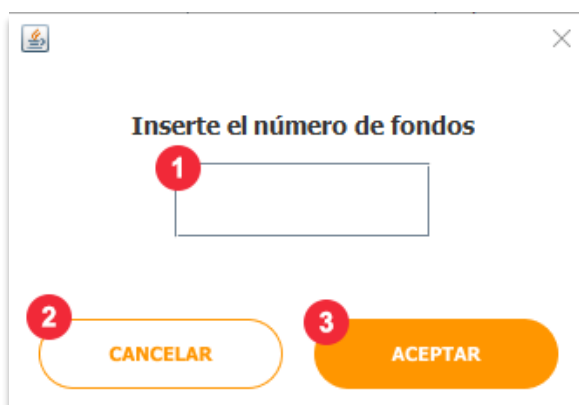
Id	Contraseña	Fondos	Tipo	Autorizado
Inditex	326212	90000	Empresa	Operativo
Repsol	326212	10000	Empresa	Operativo
Citroen	a3f38506	80000	Empresa	Dar de alta
a	a3f97	80000	Empresa	Dar de alta
Oscar	a3f25700	100	Inversor	Dar de alta
b	a3f98	100000	Inversor	Dar de alta
Diego	326212	1000000	Inversor	Operativo
Alex	a3f25186	0	Inversor	Eliminado
0	426623	1000000	Regulador	Operativo

6 **AÑADIR FONDOS** **7** **DAR DE BAJA** **8** **DAR DE ALTA** **9** **SALIR**

Se accede escribiendo las credenciales de administrador en la ventana de inicio.

1. Se escribe nombre de usuario para filtrar.
2. Filtra por el estado del usuario.
3. Filtra por el tipo de usuario.
4. Busca en la tabla, según los datos escritos en el filtro.
5. Tabla donde se muestran los datos de los usuarios existentes en la base de datos.
6. Botón que sirve para añadir fondos al usuario seleccionado [Ventana 4.19]
7. Se activa si un usuario ha solicitado que se le dé de baja, pone el saldo a 0 el usuario.
8. Se activa si un usuario ha solicitado que se le dé de alta.
9. Cierra la ventana.


4.19 Ventana Añadir Fondos



Se accede a través de la ventana principal del administrador.

1. Se inserta el valor que se quiere añadir.
2. Se cierra la ventana.
3. Suma el valor insertado a los fondos del usuario.

4.20 Cambiar Contraseña



Se accede a través del perfil de la empresa o del inversor. Su funcionalidad es cambiar la contraseña.

1. Se introduce la nueva contraseña.
2. Se introduce la nueva contraseña, otra vez.
3. Se encarga de comprobar que las dos contraseñas escritas son iguales, de serlo, modifica la contraseña.
4. Cierra la ventana.

4.21 Ventana Realizar

DATOS SOBRE LAS ACCIONES QUE POSEES

Empresa: 1 Citroen

Acciones máximas: 2 100

HACER OFERTA VENTA:

Número de acciones: 3

Precio por accion: 4

El regulador recibirá una comisión del 5%.

5 **ACEPTAR** 6 **CANCELAR**

Se accede a esta ventana a través del botón realizar oferta de venta en la Ventana principal. Es pulsado cuando hay una acción seleccionada y se interesa vender.

1. Datos de la empresa a la cual pertenece la acción.
2. Número de acciones que posee el usuario de la empresa.
3. Se inserta el número de acciones que pretende vender el usuario.
4. Precio de cada acción que pretende vender el usuario.
5. Botón para aceptar la operación, funcionará en caso de que los datos introducidos sean correctos.
6. Botón que cancela la opción y cierra la ventana.

5. Desarrollo de funcionalidades

5.1. Funcionalidades de Usuarios

[FU_01] Acceso Sistema:

Inicialmente, se accede a la aplicación con una ventana inicial [4.1 Ventana Autentificación] en la cual se solicita entrar en el sistema y con ello en la base de datos.

```
public void comprobarAutenticacion() {
    etiquetaFallo.setVisible(false);
    Usuario usuario = null;
    if (!textoID.getText().isEmpty() &&
!textoContrasena.getText().isEmpty()) {
        usuario = fa.comprobarAutenticacion(textoID.getText(),
textoContrasena.getText());
        if (usuario != null) {
            if (usuario.getEnEspera() == -3) {
                etiquetaFallo.setVisible(true);
                etiquetaFallo.setText(" Cuenta eliminada");
            } else if (usuario.getEnEspera() == 1) {
                etiquetaFallo.setVisible(true);
                etiquetaFallo.setText(" Pendiente de dar de
alta");
            } else if (usuario.getEnEspera() == -1) {
                etiquetaFallo.setVisible(true);
                etiquetaFallo.setText(" Pendiente de dar de
baja");
            } else {
                //Si el usuario es un inversor
                if (usuario.getTipoUsuario() ==
TipoUsuario.inversores) {
                    this.dispose();
                    fa.visualizarVentanaPrincipal(usuario);
                } //Si el usuario es una empresa
                else if (usuario.getTipoUsuario() ==
TipoUsuario.empresas) {
                    this.dispose();
                    fa.visualizarVentanaPrincipal(usuario);
                } //Si el usuario es el regulador
                else if (usuario.getTipoUsuario() ==
TipoUsuario.regulador) {
                    this.dispose();
                    fa.visualizarVentanaPrincipalAdmin();
                }
            }
        } else {
            etiquetaFallo.setVisible(true);
            etiquetaFallo.setText("! Datos de inicio de sesión
incorrectos");
        }
    } else {
        etiquetaFallo.setVisible(true);
        etiquetaFallo.setText("Completa todos los campos");
    }
}
```

5. Desarrollo de funcionalidades

Esta función nos traslada al DAOUsuarios donde a través de una contraseña y una idUsuario propios de la tabla Usuarios se comprueba si estos datos están en la base de datos y con ello si autenticar el usuario.

```
public Usuario validarUsuario(String idUsuario, String contrasena)
{
    Usuario resultado = null;
    Connection con;
    PreparedStatement stmUsuario = null;
    ResultSet rsUsuario;

    con = this.getConexion();

    try {
        stmUsuario = con.prepareStatement("select id_usuario,
contrasena, fondos, tipo_usuario, enespera "
+ " from usuarios "
+ " where id_usuario = ? and contrasena = ?");
        stmUsuario.setString(1, idUsuario);
        stmUsuario.setString(2, contrasena);
        rsUsuario = stmUsuario.executeQuery();
        if (rsUsuario.next()) {
            resultado = new
Usuario(rsUsuario.getString("id_usuario"),
rsUsuario.getString("contrasena"), rsUsuario.getInt("fondos"),
rsUsuario.getString("tipo_usuario"));
            resultado.setEnEspera(rsUsuario.getInt("enespera"));
        }
    } catch (SQLException e) {
        System.out.println(e.getMessage());
        switch (e.getErrorCode()) {
            case 23502:

this.getFachadaAplicacion().muestraExcepcion("Revisa aquellos campos
que deban ser no nulos.\n");
                break;
            case 23505:
                this.getFachadaAplicacion().muestraExcepcion("El
id_usuario introducido ya se encuentra en uso.\n");
                break;
            default:

this.getFachadaAplicacion().muestraExcepcion(e.getMessage());
                break;
        }
    } finally {
        try {
            stmUsuario.close();
        } catch (SQLException e) {
            System.out.println("Imposible cerrar cursores");
        }
    }
    return resultado;
}
```

Finalmente, dependiendo de si el atributo TipoUsuario de la tabla Usuarios es un inversor, empresa o regulador se trasladará a una ventana principal diferente.

[FU_02] Solicitud Registro: Una empresa o inversor solicita registrarse en el sistema en la [4.1 Ventana Autenticación] a través del botón “Registrarse”. Así se dirige al [4.2 Registro Inversores] o [4.3 Registro Empresa]

[4.2 Registro Inversores] : En este proceso de solicitud de registro, el usuario propondrá un idusuario y una clave de acceso. Además de ello, se introducirán los atributos DNI, nombre, apellido1, apellido2, calle, portal, piso, puerta, teléfonos necesarios para crear el objeto de la tabla Inversor. Se comprobará si ese objeto inversor está ya creado en la base de datos y finalmente se realizará un insert into en la tabla inversor con los datos introducidos por el usuario.

```
private void btnRegistroMouseClicked(java.awt.event.MouseEvent
evt) {
    if (!comprobarVacios()) {
        try {
            Integer.parseInt(textoPortal.getText());
            Integer.parseInt(textoTelefono.getText());
            Integer.parseInt(textoPiso.getText());
            Usuario u;
            u = new Usuario (textoID.getText(),
textoContrasena.getText(), 0, "inversores");
            Inversor i;
            i = new Inversor(u, textoDNI.getText(),
textoNombre.getText(), textoAp1.getText(), textoAp2.getText(),
textoCalle.getText(), Integer.parseInt(textoPortal.getText()),
Integer.parseInt(textoPiso.getText()), textoPuerta.getText(),
Integer.parseInt(textoTelefono.getText()));

            if (fa.registrarUsuario(u) == 0){
                if (fa.registrarInversor(i) == 0){
                    fa.muestraExcepcion("Inversor creado
correctamente.");
                    this.dispose();
                }
            }
        } catch (NumberFormatException e){
            textoError.setText("El campo portal, telefono y piso
tienen letras.");
        }
        textoError.setVisible(true);
    }
}

public int registrarInversor(Inversor i) {
    Connection con;
    PreparedStatement stmUsuario = null;

    con = super.getConexion();

    try {
        stmUsuario = con.prepareStatement("insert into
inversores(idusuario, dni, nombre, apellido1, apellido2, calle,
portal, piso, puerta, telefono) "
+ "values (?,?,?,?,?,?,?,?,?,?)");
        stmUsuario.setString(1,
i.getId_usuario().getId_usuario());
    }
```


5. Desarrollo de funcionalidades

```
stmUsuario.setString(2, i.getDNI());
stmUsuario.setString(3, i.getNombre());
stmUsuario.setString(4, i.getApellido1());
stmUsuario.setString(5, i.getApellido2());
stmUsuario.setString(6, i.getCalle());
stmUsuario.setInt(7, i.getPortal());
stmUsuario.setInt(8, i.getPiso());
stmUsuario.setString(9, i.getPuerta());
stmUsuario.setInt(10, i.getTelefono());
stmUsuario.executeUpdate();

} catch (SQLException e) {
    System.out.println(e.getMessage());
    switch (e.getErrorCode()) {
        case 23502:

this.getFachadaAplicacion().muestraExcepcion("Revisa aquellos campos
que deban ser no nulos.\n");
        break;
        case 23505:
            this.getFachadaAplicacion().muestraExcepcion("Ese
DNI ya está registrado.\n");
            break;
        default:

this.getFachadaAplicacion().muestraExcepcion(e.getMessage());
        break;
    }
    return 1;
} finally {
    try {
        stmUsuario.close();
    } catch (SQLException e) {
        System.out.println("Imposible cerrar cursores");
    }
}
return 0;//Finaliza correctamente
}
```

[4.3 Registro Empresas] : En este proceso de solicitud de registro, el usuario propondrá también un idusuario y una clave de acceso. Además de ello, se introducirán los atributos CIF, nombreComercial,ciudad,calle,número,teléfono,participaciones, participacionessin vender, necesarios para crear el objeto de la tabla Empresa (este último se inicializará automáticamente a 0). Se comprobará si ese objeto empresa está ya creado en la base de datos y finalmente se realizará un insert into en la tabla empresa con los datos introducidos por el usuario.

```
private void btnRegistroMouseClicked(java.awt.event.MouseEvent
evt) {
    int resultadoRegistro;
    if (textoCIF.getText().equals("") ||
textoContraseña.getText().equals("") ||
textoNombreEmpresa.getText().equals("") ||
textoNombreUsuario.getText().equals("") ||
textoCalle.getText().equals("") || textoPoblacion.getText().equals("")
|| textoNumero.getText().equals("")) {
```

```

        resultadoRegistro = 3;
    } else {
        try {
            Integer.parseInt(textoNumero.getText());
            Integer.parseInt(textoTelefono.getText());
            Usuario user = new
Usuario(textoNombreUsuario.getText(), textoContraseña.getText(), 0,
"empresas");
            Empresa nuevaEmpresa = new Empresa(textoCIF.getText(),
user, textoNombreEmpresa.getText(), textoPoblacion.getText(),
textoCalle.getText(), Integer.parseInt(textoNumero.getText()),
Integer.parseInt(textoTelefono.getText()), 0, 0);

            System.out.println(user.toString());
            System.out.println(nuevaEmpresa.toString());
            if (fa.registrarUsuario(user) == 0) {
                if (fa.registrarEmpresa(nuevaEmpresa) == 0) {
                    resultadoRegistro = 0;
                    fa.muestraExcepcion("Empresa creada
correctamente.");
                    this.dispose();
                }
                resultadoRegistro = 6;
            } else {
                resultadoRegistro = 5;
            }

        } catch (NumberFormatException e) {
            resultadoRegistro = 4;
        }
    }
    switch (resultadoRegistro) {
        case 0:
            etiquetaResultadoAccion.setText("Empresa registrada
correctamente!");
            break;
        case 1:
            etiquetaResultadoAccion.setText("Los datos de la
empresa no son válidos");
            break;
        case 2:
            etiquetaResultadoAccion.setText("El id de la
contraseña no es válido");
            break;
        case 3:
            etiquetaResultadoAccion.setText("Hay datos necesarios
para el registro en blanco");
            break;
        case 4:
            etiquetaResultadoAccion.setText("Alguno de los campos
teléfono y número tienes letras.");
            break;
        case 5:
            etiquetaResultadoAccion.setText("El nombre de usuario
está ocupado.");
            break;
        case 6:
            etiquetaResultadoAccion.setText("Fallo al crear la
empresa.");
            break;
        default:
    
```

5. Desarrollo de funcionalidades

```
etiquetaResultadoAccion.setText("La empresa no pudo registrarse");
        break;
    }

    etiquetaResultadoAccion.setVisible(true);
}

//Registro de empresa, inicializar participacionesSinVender
//devuelve 0 si no hay errores, 1 si hay campos nulos, 2 si
usuario repetido, 3 en otro caso
public int registrarEmpresa(Empresa empresa) {
    Connection con;
    PreparedStatement stmEmpresa = null;
    int devolver = 0;
    con = this.getConexion();

    try {
        stmEmpresa = con.prepareStatement("insert into empresas
values (?, ?, ?, ?, ?, ?, ?, 0, ?, 0)");
        stmEmpresa.setString(1,
empresa.getUsuario().getId_usuario());
        stmEmpresa.setString(2, empresa.getCIF());
        stmEmpresa.setString(3, empresa.getNombreComercial());
        stmEmpresa.setString(4, empresa.getCiudad());
        stmEmpresa.setString(5, empresa.getCalle());
        stmEmpresa.setInt(6, empresa.getNumero());
        stmEmpresa.setInt(7, empresa.getTelefono());
        stmEmpresa.setInt(8,
empresa.getParticipacionesSinVender());

        stmEmpresa.executeUpdate();
        devolver = 0;

    } catch (SQLException e) {
        System.out.println(e.getMessage());
        switch (e.getErrorCode()) {
            case 23502:
                devolver = 1;

//this.getFachadaAplicacion().muestraExcepcion("Revisa aquellos campos
que deban ser no nulos.\n");
                break;
            case 23505:
                devolver = 2;
                //this.getFachadaAplicacion().muestraExcepcion("El
id_usuario introducido ya se encuentra en uso.\n");
                break;
            default:
                devolver = 3;

//this.getFachadaAplicacion().muestraExcepcion(e.getMessage());
                break;
        }
        return 1;
    } finally {
        try {
            stmEmpresa.close();
        } catch (SQLException e) {
```

```

        System.out.println("Imposible cerrar cursores");
    }
}
return devolver;
}

```

* **[FU_03] Confirmación Registro:** El regulador entrará en el sistema a través de la [4.1 Ventana Autenticación] introduciendo su ID y contraseña. Una vez dentro de la base de datos, como una solicitud de registro se convertirá en un registro efectivo cuando el regulador lo autorice, el regulador llamará a la función DarAlta() para dar confirmar el registro de un usuario que seleccione. Más tarde, nos pasaremos al DAO para modificar el atributo enEspera de la tabla Usuario del usuario seleccionado. Este atributo tomará los siguientes valores para confirmar el registro:

- 1: Usuario pendiente de dar de alta
- 0: Usuario activo
- -1: Usuario dado de baja
- -3: Usuario eliminado

```

public void DarAlta() {
    ModeloTablaUsuarios m = (ModeloTablaUsuarios)
tablauser.getModel();
    user = m.obtenerUsuario(tablauser.getSelectedRow());
    user.setEnEspera(0);
    fa.modificarUsuario(user, user.getId_usuario());
    InicializarTabla();

    botonDarAlta.setEnabled(false);
    botonEliminar.setEnabled(false);
    BotonAnadirFondos.setEnabled(false);
}

//Modificar Usuario: Se encarga de modificar los datos del usuario
public void modificarUsuario(Usuario u, String id_usuarioantiguo)
{
    Connection con;
    PreparedStatement stmUsuario = null;

    con = super.getConexion();
    //Se entiende que los fondos y el tipo de usuario no se pueden
cambiar
    try {

        stmUsuario = con.prepareStatement("update usuarios "
            + "set contrasena=? , "
            + "    id_usuario=? , "
            + "    fondos=? , "
            + "    enespera=? "
            + "where id_usuario=?");
        stmUsuario.setString(1, u.getContrasena());
        stmUsuario.setString(2, u.getId_usuario());
        stmUsuario.setInt(3, u.getFondos());
        stmUsuario.setInt(4, u.getEnEspera());
        stmUsuario.setString(5, id_usuarioantiguo);
        stmUsuario.executeUpdate();

    } catch (SQLException e) {

```

5. Desarrollo de funcionalidades

```
        System.out.println(e.getMessage());
        switch (e.getErrorCode()) {
            case 23502:

this.getFachadaAplicacion().muestraExcepcion("Revisa aquellos campos
que deban ser no nulos.\n");
                break;
            case 23505:
                this.getFachadaAplicacion().muestraExcepcion("El
id_usuario introducido ya se encuentra en uso.\n");
                break;
            default:

this.getFachadaAplicacion().muestraExcepcion(e.getMessage());
                break;
        }
    } finally {
        try {
            stmUsuario.close();
        } catch (SQLException e) {
            System.out.println("Imposible cerrar cursores");
        }
    }
}
```

* **[FU_04] Modificación Datos:** Seleccionando la [4.4 Ventana Perfi Empresarial] o [4.5 Ventana Perfil Inversor] cualquier empresa/inversor podrá modificar sus datos de identificación en cualquier momento. Antes de modificar, se comprobará mediante un select en la tabla Empresa/Inversor si existe algún objeto con el mismo identificador propuesto en el sistema. En caso contrario, se cambiarán los campos que se llamará la función modificarUsuario() y modificarDatosInversor() las cuales harán un update en la tabla Empresa/Inversor con los atributos nuevos.

```
public void guardarDatosUsuario() {
    if (!TextoNombre.getText().isEmpty() &&
!TextoAp1.getText().isEmpty() && !TextoAp2.getText().isEmpty() &&
!TextoID.getText().isEmpty()
        && !TextoCalle.getText().isEmpty() &&
!TextoPortal.getText().isEmpty() && !TextoPiso.getText().isEmpty() &&
!TextoPuerta.getText().isEmpty() &&
!TextoTelefono.getText().isEmpty()) {
        try {
            //Comprobación errores a int
            Integer.parseInt(this.TextoTelefono.getText());
            Integer.parseInt(TextoPortal.getText());
            Integer.parseInt(TextoPiso.getText());

            Inversor newInversor = new
Inversor(fa.consultarUnUsuario(idAntiguo), TextoDNI.getText(),
TextoNombre.getText(), TextoAp1.getText(),
                TextoAp2.getText(), TextoCalle.getText(),
Integer.parseInt(TextoPortal.getText()),
Integer.parseInt(TextoPiso.getText()),
                TextoPuerta.getText(),
Integer.parseInt(TextoTelefono.getText()));

            newInversor.setApellido1(TextoAp1.getText());
```

5. Desarrollo de funcionalidades

```
newInversor.setApellido2(TextoAp2.getText());
newInversor.setCalle(TextoCalle.getText());
newInversor.setDNI(TextoDNI.getText());
newInversor.setNombre(TextoNombre.getText());

newInversor.setPiso(Integer.parseInt(TextoPiso.getText()));

newInversor.setPortal(Integer.parseInt(TextoPortal.getText()));
newInversor.setPuerta(TextoPuerta.getText());

newInversor.setTelefono(Integer.parseInt(TextoTelefono.getText()));

newInversor.getId_usuario().setId_usuario(TextoID.getText());

fa.modificarUsuario(newInversor.getId_usuario(),
idAntiguo);
fa.modificarDatosInversor(newInversor);

inversor =
fa.consultarUnInversor(fa.consultarUnUsuario(TextoID.getText()));

//Inversor newInversor = new Inversor()

fa.muestraExcepcion("Inversor modificado
correctamente.");
etiquetaResultadoAccion.setVisible(false);

} catch (NumberFormatException e) {
    etiquetaResultadoAccion.setText("Alguno de los campos
teléfono, portal y piso tienen letras.");
    etiquetaResultadoAccion.setVisible(true);
}
} else {
    etiquetaResultadoAccion.setText("Completa todos los
campos.");
    etiquetaResultadoAccion.setVisible(true);
}

}

}

//Consultar un usuario
public Usuario consultarUnUsuario(String id) {
    Usuario usuarioActual = null;
    Connection con;
    PreparedStatement stmCatalogo = null;
    ResultSet rsCatalogo;

    con = this.getConexion();

    String consulta = "select * "
        + "from usuarios as u "
        + "where id_usuario = ?";
```

5. Desarrollo de funcionalidades

```
try {
    stmCatalogo = con.prepareStatement(consulta);
    stmCatalogo.setString(1, id);

    rsCatalogo = stmCatalogo.executeQuery();

    if (rsCatalogo.next()) {
        usuarioActual = new
Usuario(rsCatalogo.getString("id_usuario"),
rsCatalogo.getString("contrasena"), rsCatalogo.getInt("fondos"),
rsCatalogo.getString("tipo_usuario"));
    }

} catch (SQLException e) {
    System.out.println(e.getMessage());
    switch (e.getErrorCode()) {
        case 23502:

this.getFachadaAplicacion().muestraExcepcion("Revisa aquellos campos
que deban ser no nulos.\n");
        break;
        case 23505:
            this.getFachadaAplicacion().muestraExcepcion("El
id_usuario introducido ya se encuentra en uso.\n");
            break;
        default:

this.getFachadaAplicacion().muestraExcepcion(e.getMessage());
        break;
    }
} finally {
    try {
        stmCatalogo.close();
    } catch (SQLException e) {
        System.out.println("Imposible cerrar cursores");
    }
}
return usuarioActual;
}

//Obtiene los datos del inversor utilizando el DNI
public Inversor consultarUnInversor(Usuario user) {
    Inversor inversorActual = null;
    Connection con;
    PreparedStatement stmCatalogo = null;
    ResultSet rsCatalogo;

    con = this.getConexion();

    String consulta = "select idusuario, dni, nombre, apellido1,
apellido2, calle, portal, piso, puerta, telefono "
        + "from inversores "
        + "where idusuario = ?";

    try {
        stmCatalogo = con.prepareStatement(consulta);
        stmCatalogo.setString(1, user.getId_usuario());
```

```

        rsCatalogo = stmCatalogo.executeQuery();
        if (rsCatalogo.next()) {
            inversorActual = new Inversor(user,
rsCatalogo.getString("dni"), rsCatalogo.getString("nombre"),
rsCatalogo.getString("apellido1"), rsCatalogo.getString("apellido2"),
rsCatalogo.getString("calle"), rsCatalogo.getInt("portal"),
rsCatalogo.getInt("piso"), rsCatalogo.getString("puerta"),
rsCatalogo.getInt("telefono"));
        }

    } catch (SQLException e) {
        System.out.println(e.getMessage());
        switch (e.getErrorCode()) {
            case 23502:

this.getFachadaAplicacion().muestraExcepcion("Revisa aquellos campos
que deban ser no nulos.\n");
                break;
            case 23505:
                this.getFachadaAplicacion().muestraExcepcion("El
id_usuario introducido ya se encuentra en uso.\n");
                break;
            default:

this.getFachadaAplicacion().muestraExcepcion(e.getMessage());
                break;
        }
    } finally {
        try {
            stmCatalogo.close();
        } catch (SQLException e) {
            System.out.println("Imposible cerrar cursores");
        }
    }
    return inversorActual;
}

public void modificarInversor(Inversor i) {
    //presupongo que el dni es inmutable
    Connection con;
    PreparedStatement stmInversor = null;

    con = super.getConexion();
    System.out.println(i.toString());
    try {
        stmInversor = con.prepareStatement("update inversores "
            + "set idusuario=? , "
            + "    nombre=? , "
            + "    apellido1=? , "
            + "    apellido2=? , "
            + "    calle=? , "
            + "    portal=? , "
            + "    piso=? , "
            + "    puerta=? , "
            + "    telefono=?, "
            + "    dni=? "
            + "    where idusuario=?");
        stmInversor.setString(1,
i.getId_usuario().getId_usuario());
        stmInversor.setString(2, i.getNombre());
    }
}

```


5. Desarrollo de funcionalidades

```
stmInversor.setString(3, i.getApellido1());
stmInversor.setString(4, i.getApellido2());
stmInversor.setString(5, i.getCalle());
stmInversor.setInt(6, i.getPortal());
stmInversor.setInt(7, i.getPiso());
stmInversor.setString(8, i.getPuerta());
stmInversor.setInt(9, i.getTelefono());
stmInversor.setString(10, i.getDNI());
stmInversor.setString(11,
i.getId_usuario().getId_usuario());
stmInversor.executeUpdate();

} catch (SQLException e) {
    System.out.println(e.getMessage());
    switch (e.getErrorCode()) {
        case 23502:

this.getFachadaAplicacion().muestraExcepcion("Revisa aquellos campos
que deban ser no nulos.\n");
        break;
        case 23505:
            this.getFachadaAplicacion().muestraExcepcion("El
id_usuario introducido ya se encuentra en uso.\n");
            break;
        default:

this.getFachadaAplicacion().muestraExcepcion(e.getMessage());
        break;
    }
} finally {
    try {
        stmInversor.close();
    } catch (SQLException e) {
        System.out.println("Imposible cerrar cursores");
    }
}
}

//Modificar Usuario: Se encarga de modificar los datos del usuario
public void modificarUsuario(Usuario u, String id_usuarioantiguo)
{
    Connection con;
    PreparedStatement stmUsuario = null;

    con = super.getConexion();
    //Se entiende que los fondos y el tipo de usuario no se pueden
    cambiar
    try {

        stmUsuario = con.prepareStatement("update usuarios "
            + "set contrasena=? , "
            + "    id_usuario=? , "
            + "    fondos=? , "
            + "    enespera=? "
            + "where id_usuario=?");
        stmUsuario.setString(1, u.getContrasena());
        stmUsuario.setString(2, u.getId_usuario());
        stmUsuario.setInt(3, u.getFondos());
        stmUsuario.setInt(4, u.getEnEspera());
        stmUsuario.setString(5, id_usuarioantiguo);
```

```

stmUsuario.executeUpdate();

} catch (SQLException e) {
    System.out.println(e.getMessage());
    switch (e.getErrorCode()) {
        case 23502:

this.getFachadaAplicacion().muestraExcepcion("Revisa aquellos campos
que deban ser no nulos.\n");
        break;
        case 23505:
            this.getFachadaAplicacion().muestraExcepcion("El
id_usuario introducido ya se encuentra en uso.\n");
            break;
        default:

this.getFachadaAplicacion().muestraExcepcion(e.getMessage());
        break;
    }
} finally {
    try {
        stmUsuario.close();
    } catch (SQLException e) {
        System.out.println("Imposible cerrar cursores");
    }
}
}
}

```

* **[FU_05] Modificación Saldo Cuenta:** En la [Ventana Administrador], el regulador seleccionará el BotonAñadirFondos a un Usuario de la tabla.

```

private void
BotonAnadirFondosActionPerformed(java.awt.event.ActionEvent evt) {
    ModeloTablaUsuarios m = (ModeloTablaUsuarios)
tablauser.getModel();
    this.user = m.obtenerUsuario(tablauser.getSelectedRow());
    if(user != null){
        ModificarFondo();
    }
}

public Usuario obtenerUsuario(int index){
    return this.usuarios.get(index);
}

```

Si el usuario seleccionado de la tabla no es nulo, se traslada a [Ventana Añadir Fondos] en la cual se introduce una cantidad (se comprueba que es un int) y se llama una vez mas a la función ModificarUsuario() para introducir los fondos.

```

public void ModificarFondo(){
    ModeloTablaUsuarios m = (ModeloTablaUsuarios)
tablauser.getModel();
    user = m.obtenerUsuario(tablauser.getSelectedRow());

    IVentana iVentana = new IVentana() {
        @Override
        public void onSuccess() {

```

5. Desarrollo de funcionalidades

```
InicializarTabla();
BotonAnadirFondos.setEnabled(false);
botonDarAlta.setEnabled(false);
botonEliminar.setEnabled(false);
    }
};

fa.visualizarVModificarFondos(user, iVentana);
}

private void btnAceptarMouseClicked(java.awt.event.MouseEvent evt)
{
    if(!TextoFondos.getText().isEmpty()){
        try{
            parseInt(TextoFondos.getText());

user.setFondos(user.getFondos()+parseInt(TextoFondos.getText()));
            fa.modificarUsuario(user, user.getId_usuario());
            iVentana.onSuccess();
            this.dispose();
        } catch (NumberFormatException e) {
            fa.muestraExcepcion("Introduce un número");
        }
    }
}
```

* **[FU_06] Solicitud Baja:** Para solicitar una baja de cualquier empresa/inversor, se establecerá el valor del atributo enEspera de la tabla Usuarios a -1. Este valor indicará al sistema que el usuario esta dado de baja. Finalmente, modificamos este valor en la base de datos a través de la función ModificarUsuario() escrita en anteriores funcionalides() mediante un update usuarios set.

```
private void btnDarBajaMouseClicked(java.awt.event.MouseEvent evt)
{
    Usuario user = inversor.getId_usuario();
    user.setEnEspera(-1);
    fa.modificarUsuario(user, user.getId_usuario());
    this.dispose();
    fa.iniciaInterfazUsuario();
}
```

* **[FU_07] Confirmación Baja** En la [Ventana Principal Administrador] nos encontramos con el botón “Dar de baja”. De esta manera, el regulador confirma la baja de un usuario llamando a la función EliminarAccion(). En el caso de que el saldo no este a cero, se hace un setter en el atributo Fondos de la tabla Usuario y pasaremos el atributo EnEspera de la tabla Usuarios a -3 para indicar en nuestro sistema que la baja ha sido tramitada. Finalmente, llamamos a la función ModificarUsuario(). Esta esta escrita en anteriores transacciones y modificar estos atributos de la tabla Usuarios en la base de datos

```
public void EliminarAccion(){
```

```

        ModeloTablaUsuarios m = (ModeloTablaUsuarios)
tablauser.getModel();
        user = m.obtenerUsuario(tablauser.getSelectedRow());
        user.setEnEspera(-3);
        user.setFondos(0);
        fa.modificarUsuario(user, user.getId_usuario());
        InicializarTabla();

        botonDarAlta.setEnabled(false);
        botonEliminar.setEnabled(false);
        BotonAnadirFondos.setEnabled(false);
    }

```

5. 2. Funcionalidades de comercio

* **[FC_01] Alta Participaciones:** Las empresas registradas podrán dar de alta el número de participaciones que deseen. Las participaciones dadas de alta por una empresa se incorporarán a su propia cartera de participaciones. Por simplicidad del sistema, las participaciones no se identificarán de forma unitaria. Sólo se registrará la cantidad total de participaciones, de cada empresa, que posee cada empresa/inversor y el número total de participaciones involucradas en cada operación de compra/venta.

Se implementa mediante la transacción generarAcciones en DAOEmpresas, actualizando la tabla poseer.

```

public int GenerarAcciones(Usuario usuario, int cantidad) {

        Empresa emp = this.consultarEmpresas
(usuario.getId_usuario())        .get(0);

        Connection con;

        PreparedStatement stmUsuario = null;

        con = super.getConexion();

        try {

                stmUsuario = con.prepareStatement("insert into
poseer(empresa,          idempresa, idusuario, fecha, cantidad ) "
+ "values                (?,?,,CURRENT_TIMESTAMP,?)" );

                stmUsuario.setString(1, emp.getCIF());

                stmUsuario.setString(2, usuario.getId_usuario());

```

5. Desarrollo de funcionalidades

```
stmUsuario.setString(3, usuario.getId_usuario());

stmUsuario.setInt(4, cantidad);

stmUsuario.executeUpdate();          }

    catch (SQLException e) {

        System.out.println(e.getMessage()); return 1;
    }

    finally {

        try {                stmUsuario.close();                }

        catch (SQLException e) {

            System.out.println("Imposible cerrar cursores");                }
        return 0;    }
}
```

* **[FC_02] Baja Participaciones:** Las empresas registradas podrán dar de baja el número de participaciones que deseen, siempre y cuando dispongan de ellas en su propia cartera. Las participaciones dadas de baja por una empresa se eliminarán del sistema disminuyendo el número total de participaciones existentes.

Se implementa mediante la transacción retirarAcciones en DAOCompraventa, actualizando la tabla poseer.

```
public void retirarParticipaciones(String iduser, int cantidad, String
cif){

    Connection con;

    PreparedStatement stmUsuario = null;

    con = super.getConexion();

    String consulta= "UPDATE poseer SET cantidad = cantidad - ?
    where idusuario = ?";

    try {

        stmUsuario = con.prepareStatement(consulta);
```

```

        stmUsuario.setInt(1, cantidad);

        stmUsuario.setString(2, iduser);

        stmUsuario.executeUpdate();

    } catch (SQLException e) {

        System.out.println(e.getMessage());

    }

    finally {

        try {

            stmUsuario.close();

        } catch (SQLException e) {

            System.out.println("Imposible cerrar cursores");

        }

    }

}

```

***[FC_03]Comisión Compra/Venta:** El regulador del mercado fijara una comisión sobre las operaciones de compra/venta. El importe de la comisión de compra/venta deberá ser público y se informará al vendedor en el momento de realizar la oferta de venta. El importe de dicha comisión, a deducir del ingreso al vendedor, será ingresado en la cuenta especial del regulador. La comisión a aplicar a una operación de compra/venta será la disponible de forma pública en el sistema en el momento de poner en el mercado la oferta de venta.

Se ha fijado una comisión del 5% para todas las transacciones, atributo que se ha fijado desde el DAOCompraventa. Además, esta comisión será visible en la tabla de compraventa desde la interfaz. Se ha aplicado el comando `setAutocommit(False)` para permitir hacer un `rollback()` en caso de error.

```

public Boolean Insertartuplacompraventa(String comprador, String
vendedor, String cifempresa, String idempresa, int cantidad) {
    Connection con;
    PreparedStatement stmUsuario = null;
    Boolean resultado = false;
    con = super.getConexion();

    try {
        con.setAutoCommit(false);

```

5. Desarrollo de funcionalidades

```
stmUsuario = con.prepareStatement("insert into compraventa
(idusuariovendedor, idusuariocomprador, empresa, idempresa,
fechacompra, numeroparticipaciones, comision ) values
(?,?,?,?, CURRENT_TIMESTAMP, ?, 0.05)");
stmUsuario.setString(1, vendedor);
stmUsuario.setString(2, comprador);
stmUsuario.setString(3, cifempresa);
stmUsuario.setString(4, idempresa);
stmUsuario.setInt(5, cantidad);

int affectedrows = stmUsuario.executeUpdate();
if (affectedrows > 0) {
    resultado = true;
} else {
    con.rollback();
}

} catch (SQLException e) {
    System.out.println(e.getMessage());
}
try {
    if (con != null) {
        con.rollback();
    }
} catch (SQLException er) {
    System.out.println(er.getMessage());
}
return false;
} finally {
    try {
        stmUsuario.close();
    } catch (SQLException e) {
        System.out.println("Imposible
        cerrar cursores");
    }
}
return resultado;
}
```

* **[FC_04] Venta Participaciones:** Cualquier usuario, inversor y/o empresa, poseedor de participaciones de alguna empresa podrá ponerlas a la venta. El total de participaciones puestas a la venta por un usuario no puede ser superior al número de participaciones que posee dicho usuario. Para ello deberá indicar la empresa, el número global de participaciones y el precio de venta. Cuando las participaciones se vendan, recibirá en su cuenta el importe de la venta menos la comisión fijada por el mercado de valores. Las ofertas de venta permanecerán en el mercado de forma permanente salvo que sean retiradas por el usuario que las realizo.

Se ha implementado mediante la realización de la transacción crear insertarOferta actualizando la tabla de hacerOfertaVenta. Se ha establecido, como en casos anteriores, un contador de tuplas afectadas para poder considerar si la transacción ha tenido éxito, así como un autocommit(false) para permitir deshacer la transacción en caso de error con rollback().

```

public Boolean insertarOferta(HacerOfertaVenta oferta) {

    Connection con;

    PreparedStatement stmUsuario = null;

    Boolean resultado=false;

    ResultSet rsCatalogo;          con = super.getConexion();

    try {

        con.setAutoCommit(false);

        String consulta = "select * from poseer where idusuario=?
        and idempresa=? and ?<=cantidad-(select COALESCE(sum
        (numparticipaciones),0) from haceroverta where
        idusuario=? and idempresa=? )";

        stmUsuario = con.prepareStatement(consulta);

        stmUsuario.setString(1,oferta.getUsuario().
        getId_usuario());

        stmUsuario.setString(2,
        oferta.getEmpresa().getUsuario().getId_usuario());

        stmUsuario.setInt(3, oferta.getNumeroParticipaciones());

        stmUsuario.setString(4, oferta.getUsuario().
        getId_usuario());

        stmUsuario.setString(5, oferta.getEmpresa().getUsuario().
        getId_usuario());

        rsCatalogo = stmUsuario.executeQuery();

        if(rsCatalogo.next()){

```


5. Desarrollo de funcionalidades

```
con.setAutoCommit(false);

stmUsuario = con.prepareStatement("insert into
Hacerofertaventa (fechaoferta,precio,numparticipaciones,
disponibilidad,empresa,idempresa,idusuario ) "
+
"values (CURRENT_TIMESTAMP, ?,?,true,?,?,? )");

stmUsuario.setFloat(1,
oferta.getPrecio());

stmUsuario.setInt(2,
oferta.getNumeroParticipaciones());

stmUsuario.setString(3,
oferta.getEmpresa().getCIF());

stmUsuario.setString(4,
oferta.getEmpresa().getUsuario().
getId_usuario());

stmUsuario.setString(5, oferta.getUsuario().
getId_usuario());

int
affectedrows=stmUsuario.executeUpdate();
if(affectedrows>0){                                resultado=true;
}

else{      con.rollback();  }                      else{

fa.muestraExcepcion("No posees suficientes acciones para
ofertar esa cantidad.");          }          } catch
(SQLException e) {
System.out.println(e.getMessage());

}          try {                                if
(con != null) {                                con.rollback();
}          } catch (SQLException er) {
System.out.println(er.getMessage());          }
return false;          } finally {          try {
con.commit();
stmUsuario.close();          } catch (SQLException e) {
```

```

        System.out.println("Imposible cerrar cursores");
    }
}
return resultado;
}

```

* **[FC_05] Baja Venta Participaciones:** El usuario que ha realizado una oferta de venta podrá eliminarla en cualquier momento. En el caso de ejecuciones parciales de la oferta de venta, la baja afectará a la parte de la oferta no ejecutada.

Se ha implementado mediante la transacción borrarOfertaVenta:

```

public java.util.List<HacerOfertaVenta> buscarOferta(java.util.Date
fecha, String cif, String idempresa, String idusuario) {

```

```

    java.util.List<HacerOfertaVenta> resultado = new
    java.util.ArrayList<HacerOfertaVenta>();

    HacerOfertaVenta ofertaActual;

    Connection con;

    PreparedStatement stmCatalogo = null;

    ResultSet rsCatalogo;

    con = this.getConexion();

    String consulta = "select * " + "from
    hacerofertaventa " +
    "where fechaoferta = ? and empresa = ? and idempresa = ? And
    idusuario = ? ";

    try {

        stmCatalogo = con.prepareStatement(consulta);

        stmCatalogo.setTimestamp(1, (Timestamp) fecha);

        stmCatalogo.setString(2, cif);
    }
}

```

5. Desarrollo de funcionalidades

```
stmCatalogo.setString(3, idempresa);

stmCatalogo.setString(4, idusuario);

rsCatalogo = stmCatalogo.executeQuery();

while (rsCatalogo.next()) {

    Usuario u =
this.getFachadaAplicacion().consultarUsuarios(

    rsCatalogo.getString("idusuario")).get(0);

    ofertaActual = new HacerOfertaVenta(rsCatalogo.getTimestamp

    ("fechaoferta"), rsCatalogo.getFloat("precio"),

    rsCatalogo.getBoolean("disponibilidad"),

    this.getFachadaAplicacion().consultarEmpresas(rsCatalogo.

    getString("idempresa")).get(0), u, rsCatalogo.getInt

    ("numparticipaciones"));

    resultado.add(ofertaActual);          }          } catch
(SQLException e) {
System.out.println(e.getMessage());

    } finally {          try {
stmCatalogo.close();          } catch (SQLException e) {
System.out.println("Imposible cerrar cursores");          }

}          return resultado;    }
```

* **[FC_06] Compra Participaciones:** Cualquier usuario podrá realizar ofertas de compra, siempre y cuando disponga de saldo suficiente en su cuenta. Para ello deberá indicar la empresa, el número global de participaciones y el precio máximo de compra. La compra se realizará de forma completa si hay suficientes participaciones a la venta al precio indicado o inferior (una compra puede completarse a partir de varias ofertas de venta; en ese caso, se seleccionaran las ofertas de venta por orden de precio y de antigüedad) o de forma parcial si hay menos participaciones a la venta, al precio fijado o inferior, de las indicadas en la orden de compra. Las ordenes de compra no quedarán en el mercado una vez cruzadas con las ofertas de venta existentes, tanto se hayan realizado de forma parcial o, incluso, no se hayan podido realizar por no existir oferta adecuada.

Como ya se ha indicado en **[FC_03]**, se ha empleado la siguiente transacción:

```

public Boolean Insertartuplacompraventa(String comprador, String
vendedor, String cifempresa, String idempresa, int cantidad) {
    Connection con;
    PreparedStatement stmUsuario = null;
    Boolean resultado = false;
    con = super.getConexion();

    try {
        con.setAutoCommit(false);
        stmUsuario = con.prepareStatement("insert into compraventa
(idusuariovendedor, idusuariocomprador, empresa, idempresa,
fechacompra, numeroparticipaciones, comision ) values
(?,?,?,?,?, CURRENT_TIMESTAMP, ?, 0.05)");
        stmUsuario.setString(1, vendedor);
        stmUsuario.setString(2, comprador);
        stmUsuario.setString(3, cifempresa);
        stmUsuario.setString(4, idempresa);
        stmUsuario.setInt(5, cantidad);

        int affectedrows = stmUsuario.executeUpdate();
        if (affectedrows > 0) {
            resultado = true;
        } else {
            con.rollback();
        }

    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
    try {
        if (con != null) {
            con.rollback();
        }
        catch (SQLException er) {
            System.out.println(er.getMessage());
        }
        return false;
    } finally {
        try {
            stmUsuario.close();
        } catch (SQLException e) {
            System.out.println("Imposible
cerrar cursores");
        }
    }
    return resultado;
}

```

* **[FC_07] Alta Pago Beneficios:** Desde la [Ventana Principal] se puede hacer esta funcionalidad clickando el botón de Crear Anuncios o desde [Ventana Mi Empresa] en el desplegable “Mis Anuncios”. Este botón desplaza a la [Ventana Crear Anuncio] y llamando a la función

5. Desarrollo de funcionalidades

CrearAnuncio() que nos traslada al DAOEmpresas con los parámetros del CIF empresa, la fecha de creación y el nombre del propio anuncio. Aquí se hace un insert into en la tabla Anuncios.

```
public void crearAnuncio() {
    if (!informacion.getText().isEmpty() &&
        datePicker1.getDateTimePermissive() != null) {
        try {
            Date date;
            Date dataActual = new Date();

            date =
Date.from(datePicker1.getDateTimePermissive().toInstant(ZoneOffset
.UTC));

            if (date.after(dataActual)) {
                fa.crearAnuncio(empresa.getCIF(), date,
informacion.getText());

                fa.muestraExcepcion("Anuncio creado correctamente
correctamente.");
                this.dispose();

            } else {
                fa.muestraExcepcion("Fecha introducida no valida.
Introduzca una fecha posterior a la actual.");
            }
        } catch (NumberFormatException e) {
            fa.muestraExcepcion("Alguno de los campos contienen
datos no válidos.");
        }
    } else {
        fa.muestraExcepcion("Completa todos los campos.");
    }
}
```

```
public int crearAnuncio(String cif, Date fechaPago, String
informacion) {
    Connection con;
    PreparedStatement stmEmpresa = null;
    int devolver = 0;
    con = this.getConexion();
    ResultSet resul;

    int id=0;

    try {
        stmEmpresa = con.prepareStatement("select idanuncio from
anuncios");
        resul = stmEmpresa.executeQuery();

        while(resul.next()){
            id = resul.getInt("idanuncio");
        }
    } catch (SQLException e) {
        return 1;
    }

    id++;

    try {
```

```

        stmEmpresa = con.prepareStatement("insert into anuncios
(idanuncio, informacion, fechapubli, fechapago, empresa) values (?, ?,
CURRENT_TIMESTAMP, ?, ?)");

        stmEmpresa.setInt(1, id);
        stmEmpresa.setString(2, informacion);
        stmEmpresa.setTimestamp(3, new
Timestamp(fechaPago.getTime()));
        stmEmpresa.setString(4, cif);

        stmEmpresa.executeUpdate();

        devolver = 0;

    } catch (SQLException e) {
        return 1;
    } finally {
        try {
            stmEmpresa.close();
        } catch (SQLException e) {
            System.out.println("Imposible cerrar cursores");
        }
    }
    return devolver;
}

```

* **[FC_08] Baja Pago Beneficios:** En el desplegable “Mis anuncios” de la [Ventana Mi Empresa], se seleccionaría el botón “Borrar Anuncio”. Para ello se debe tener una fila seleccionada y con ello un anuncio. Cuando se seleccione un anuncio, se introduce el id de este en la función borrarAnuncio() del DAO. Con ello, se hace un delete en la tabla de anuncios que tenga el id mencionado.

```

public void borrarAnuncio(int id) {
    Connection con;
    PreparedStatement stmEmpresa = null;

    con = this.getConexion();

    try {
        stmEmpresa = con.prepareStatement("delete from anuncios
where idanuncio = ?");

        stmEmpresa.setInt(1, id);

        stmEmpresa.executeUpdate();

    } catch (SQLException e) {
        System.out.println(e.getMessage());

        //this.getFachadaAplicacion().muestraExcepcion(e.getMessage());
    } finally {
        try {
            stmEmpresa.close();
        } catch (SQLException e) {
            System.out.println("Imposible cerrar cursores");
        }
    }
}

```

5. Desarrollo de funcionalidades

```
}
```

La eliminación de información sobre pagos de beneficios sólo podrá ser realizada por el regulador del mercado de valores a petición de la empresa que lo ha realizado (en la base de datos no se almacenará ninguna información sobre dichas peticiones de baja).

* **[FC_09] Pago Beneficios:** En el desplegable “Mis anuncios” de la [Ventana Mi Empresa], se seleccionaría el botón “Realizar pago”. Seleccionando un anuncio(fila) de la tabla, nos desplazamos a la [Ventana Pago Por acción]. Aquí, introducimos el precio que a pagar de las participaciones y la empresa a la cual vamos a realizar la funcionalidad. Finalmente, pagamos las participaciones en nuestro sistema a aquellos que tienen el atributo EnEspera=0 de la tabla Usuarios.

Si un pago dividendo está anunciado la variable opción va a estar 1, y por lo tanto se eliminará el anuncio asociado a dicho pago. Si realiza un pago sin ser anunciado previamente, la variable opción estará a 0 y no habrá anuncio que borrar.

```
private void btnRealizarPagoMouseClicked(java.awt.event.MouseEvent
evt) {

    int filaSeleccionada = tablaAnuncios.getSelectedRow();
    fa.visualizarPagoParticipaciones(empresa, 1,
Integer.parseInt(tablaAnuncios.getValueAt(filaSeleccionada,
0).toString()));

    actualizarTabla("");
    fa.muestraExcepcion("Anuncio borrado correctamente");
}

private void btnAceptarMouseClicked(java.awt.event.MouseEvent evt)
{
    int precio;
    try{
        precio = Integer.parseInt(textoPrecio.getText());
        if(this.fa.pagarParticipaciones(precio, empresa)){
            if(opcion == 1){
                this.fa.borrarAnuncio(idanuncio);
            }
            this.dispose();
            this.fa.muestraExcepcion("Pago realizado con exito");
        }else{
            this.dispose();
        }
    }catch(NumberFormatException public boolean
pagarParticipaciones(int precioAccion, Empresa empresa) {
    Connection con;
    PreparedStatement stmEmpresa = null;
    PreparedStatement stmPago = null;
    PreparedStatement stmAccion = null;
    PreparedStatement stmFondosEmpresa = null;
```

```

    ResultSet rsAccion;
    ResultSet rsEmpresa;

    con = this.getConnection();

    int affectedrows = 0;

    try {
        con.setAutoCommit(false);

        stmEmpresa = con.prepareStatement("select * from usuarios
" +
                                           " where id_usuario =
?" +
                                           " and fondos >=
(select sum(cantidad)* ?" +
                                           " from poseer" +
                                           " where idempresa = ?"
+
                                           " and idusuario != ?"
+
                                           " and idusuario
in(select id_usuario from usuarios where enespera = 0))");
        stmEmpresa.setString(1,
empresa.getUsuario().getId_usuario());
        stmEmpresa.setInt(2, precioAccion);
        stmEmpresa.setString(3,
empresa.getUsuario().getId_usuario());
        stmEmpresa.setString(4,
empresa.getUsuario().getId_usuario());

        rsEmpresa = stmEmpresa.executeQuery();

        if (rsEmpresa.next()) {
        } else {
            this.getFachadaAplicacion().muestraExcepcion("No se
pudo modificar fondos vendedor. La empresa no tiene fondos suficientes
para pagar");
            con.rollback();
            return false;
        }

        stmAccion = con.prepareStatement("select idusuario,
cantidad from poseer where idempresa = ?" +
                                           " and idusuario != ?"
+
                                           " and idusuario in
(select id_usuario from usuarios where enespera = 0))");
        stmAccion.setString(1,
empresa.getUsuario().getId_usuario());
        stmAccion.setString(2,
empresa.getUsuario().getId_usuario());

        rsAccion = stmAccion.executeQuery();

        while (rsAccion.next()) {
            stmPago = con.prepareStatement("update usuarios "
            + "set fondos= fondos + ? "

```


5. Desarrollo de funcionalidades

```
        + "where id_usuario = ?");

        stmPago.setInt(1, rsAccion.getInt("cantidad") *
precioAccion);
        stmPago.setString(2, rsAccion.getString("idusuario"));

        affectedrows = stmPago.executeUpdate();
        if (affectedrows > 0) {
        } else {
            this.getFachadaAplicacion().muestraExcepcion("No
se pudo modificar fondos vendedor");
            con.rollback();
            return false;
        }

        stmFondosEmpresa = con.prepareStatement("update
usuarios "
            + "set fondos= fondos - ? "
            + "where id_usuario = ?");

        stmFondosEmpresa.setInt(1, rsAccion.getInt("cantidad")
* precioAccion);
        stmFondosEmpresa.setString(2,
empresa.getUsuario().getId_usuario());

        affectedrows = stmFondosEmpresa.executeUpdate();
        if (affectedrows > 0) {
        } else {
            this.getFachadaAplicacion().muestraExcepcion("No
se pudo modificar fondos vendedor");
            con.rollback();
            return false;
        }
    }

} catch (SQLException e) {
    System.out.println(e.getMessage());

    try {
        if (con != null) {
            con.rollback();
        }
    } catch (SQLException er) {
        System.out.println(er.getMessage());
    }
    return false;
} finally {
    try {
        con.commit();
        stmEmpresa.close();
    } catch (SQLException e) {
        System.out.println("Imposible cerrar cursores");
    }
}
return true;
}
n e){
```

```

        this.fa.muestraExcepcion("Introduzca un número");
    }
}

```

5.3. Funcionalidades adicionales

[FEXTRA_01] : Criptomonedas: Desde la [Ventana Principal] se selecciona la opción de Criptomonedas. En ella se puede observar la existencia de una tabla en la que se guardan las diferentes criptomonedas que tiene un usuario junto a su cantidad y al precio por criptomoneda. Además, existen 3 botones de los cuales el botón volver sirve para acceder a la [Ventana Principal]. El segundo botón sirve para ir a la [Ventana compra criptomonedas] cuyo contenido explicaremos a posteriori. Por último, encontramos un botón de venta de criptomonedas cuya funcionalidad es coger la fila seleccionada y poder vender todas o parte de las criptomonedas que tiene un usuario. Esta última funcionalidad estará visible en [Ventana oferta criptomonedas].

Venta de criptomonedas

Una vez que se hace click en la oferta de criptomonedas, se lee la fila que está seleccionada y se envía a la ventana oferta criptomonedas la información acerca del usuario y del tipo de criptomoneda. A continuación, en la ventana oferta criptomoneda, se introducirá por pantalla una cantidad mayor que cero y menor que la cantidad máxima.

Por último, al pulsar sobre el botón de aceptar, se insertará oferta en la tabla de ofertas y se reducirá la cantidad de criptomonedas en posesión del usuario.

```

private void btnVenderMouseClicked(java.awt.event.MouseEvent evt) {
    int seleccion = jTable1.getSelectedRow();
    if (jTable1.getSelectedRow() != -1) {
        String nombreCripto = jTable1.getValueAt(seleccion,
0).toString();
        float cantidad =
Float.parseFloat(jTable1.getValueAt(seleccion, 1).toString());
        String tipo = jTable1.getValueAt(seleccion, 1).toString();
        IVentana iventana = new IVentana() {
            @Override
            public void onSuccess() {
                actualizarTabla("");
            }
        };
        fa.visualizarVVenderCripto(usuario, nombreCripto,
cantidad, iventana);
    }
}

private void botonVenderActionPerformed(java.awt.event.ActionEvent
evt) {
    try {
        float cantidadVender =
Float.parseFloat(criptosAVender.getText());
        if (cantidad >= cantidadVender && cantidadVender > 0) {

```

5. Desarrollo de funcionalidades

```
        boolean i =
fa.crearOfertaVentaCRIPTOMONEDA(usuario.getId_usuario(), nombrecripto,
cantidadVender);
        this.dispose();
        ventana.onSuccess();
    } else {
        fa.muestraExcepcion("Número incorrecto. Deberá ser
mayor que 0 y menor o igual que la cantidad máxima.");
    }
} catch (Exception e) {
    fa.muestraExcepcion("Valores introducidos incorrectos.");
}
}
```

```
public boolean crearOfertaVentaCRIPTOMONEDA(String usuario, String
nombreCripto, float cantidadAVender) {
    boolean toReturn = false;
    Connection con;
    PreparedStatement stmConsultaCRIPTOMONEDAS = null;
    ResultSet resultadoOperacion;
    Criptomoneda auxiliar;

    con = this.getConexion();
    System.out.println("1");
    String consulta = "insert into haceroferaventacripto values
(current_timestamp,?,?,?)";
    String consulta2 = "UPDATE monederos SET cantidad = cantidad -
? where id_usuario = ? and tipo = ?";
    try {
        stmConsultaCRIPTOMONEDAS = con.prepareStatement(consulta);
        stmConsultaCRIPTOMONEDAS.setFloat(1, cantidadAVender);
        stmConsultaCRIPTOMONEDAS.setString(2, nombreCripto);
        stmConsultaCRIPTOMONEDAS.setString(3, usuario);
        stmConsultaCRIPTOMONEDAS.executeUpdate();

        stmConsultaCRIPTOMONEDAS =
con.prepareStatement(consulta2);
        stmConsultaCRIPTOMONEDAS.setFloat(1, cantidadAVender);
        stmConsultaCRIPTOMONEDAS.setString(2, usuario);
        stmConsultaCRIPTOMONEDAS.setString(3, nombreCripto);
        stmConsultaCRIPTOMONEDAS.executeUpdate();

    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }

    this.getFachadaAplicacion().muestraExcepcion(e.getMessage());
    return false;
} finally {
    try {
        stmConsultaCRIPTOMONEDAS.close();
    } catch (SQLException e) {
        System.out.println("Imposible cerrar cursores");
    }
}

return true;
}
```

Compra de criptomonedas

[Ventana Compra Criptos] se visualizará en una tabla con todas las criptomonedas disponibles para comprar. Seleccionada una fila de la tabla, se creará un objeto oferta venta con las características de esta fila. Se comprobará si el atributo cantidad es mayor que 0, y si es afirmativo, se llamará a la función ComprarCripto() situada el DAOCriptoMonedas.

```
private void botonCompraMouseClicked(java.awt.event.MouseEvent evt) {

    IVentana iventana = new IVentana() {
        @Override
        public void onSuccess() {
            actualizarTabla("");
        }
    };
    fa.visualizarCompraCriptos(usuario, iventana);
}

private void botonComprarMouseClicked(java.awt.event.MouseEvent
evt) {
    // TODO add your handling code here:
    if (numeroCompra.getText() != null) {
        int Seleccion = jTable1.getSelectedRow();

        HacerOfertaVentaCripto oferta = new
HacerOfertaVentaCripto(jTable1.getValueAt(Seleccion, 0).toString(),
jTable1.getValueAt(Seleccion, 1).toString(),
(Date) jTable1.getValueAt(Seleccion, 4),
Float.parseFloat(jTable1.getValueAt(Seleccion,
2).toString()),
Float.parseFloat(jTable1.getValueAt(Seleccion,
3).toString()));

        try {

            Float cantidad =
Float.parseFloat(numeroCompra.getText());
            if(cantidad>0){
                fa.ComprarCriptos(usuario,
Float.parseFloat(numeroCompra.getText()), oferta);
                this.dispose();
            }else{
                fa.muestraExcepcion("Introduce un
número positivo");
            }
        } catch (NumberFormatException e) {
            fa.muestraExcepcion("Introduce un número");
        }
        ventana.onSuccess();
    } else {
        fa.muestraExcepcion("Introduce una cantidad");
    }

}

public Boolean ComprarCriptos(Usuario u, float cantidad,
HacerOfertaVentaCripto oferta) {
```

5. Desarrollo de funcionalidades

```
Boolean resultado = false;
Connection con;

PreparedStatement stmCatalogo = null;
ResultSet rsCatalogo;
con = this.getConexion();

try {
    con.setAutoCommit(false);

    //Primeira comprobación
    String consulta = "select precio          fom
hacerofertaventacripto, tablaactualizacionespacios where idusuario=?
and fechooferta=? and tipo = ? and tipo = nombrecripto and
(precio*?)<= (select fondos from usuarios where id_usuario = ?) and
cantidad >= ?";
    stmCatalogo = con.prepareStatement(consulta);
    stmCatalogo.setString(1, oferta.getId_usuario());
    stmCatalogo.setTimestamp(2, (Timestamp)
oferta.getFechaPublicacion());
    stmCatalogo.setString(3, oferta.getTipo());
    stmCatalogo.setFloat(4, cantidad);
    stmCatalogo.setString(5, u.getId_usuario());
    stmCatalogo.setFloat(6, cantidad);

    rsCatalogo = stmCatalogo.executeQuery();
    if (rsCatalogo.next()) {
        resultado = true;
    } else {
        this.getFachadaAplicacion().muestraExcepcion("Los
fondos no son suficientes o la oferta ya ha sido comprada.");
        con.rollback();
        return false;
    }

    PreparedStatement stmUsuario = null;
    //Insertar tupla compraventa

    //Comprobación cantidad igual a oferta
    consulta = "select precio from hacerofertaventacripto,
tablaactualizacionespacios "
        + "where tipo=nombrecripto and tipo=? and
fechooferta=? and idusuario= ? and cantidad=? ";
    stmCatalogo = con.prepareStatement(consulta);
    stmCatalogo.setString(1, oferta.getTipo());
    stmCatalogo.setTimestamp(2, (Timestamp)
oferta.getFechaPublicacion());
    stmCatalogo.setString(3, oferta.getId_usuario());
    stmCatalogo.setFloat(4, cantidad);
    rsCatalogo = stmCatalogo.executeQuery();
    if (rsCatalogo.next()) { //igual
        //Borrar oferta
        stmUsuario = con.prepareStatement("delete from
hacerofertaventacripto where idusuario=? and tipo= ? and
fechooferta=?");
        stmUsuario.setString(1, oferta.getId_usuario());
        stmUsuario.setString(2, oferta.getTipo());

        stmUsuario.setTimestamp(3, (Timestamp)
oferta.getFechaPublicacion());
        int affectedrows = stmUsuario.executeUpdate();
    }
}
```

```

        if (affectedrows > 0) {
        } else {
            this.getFachadaAplicacion().muestraExcepcion("No
se pudo borrar correctamente la oferta");
            con.rollback();
            return false;
        }

        //Borrar poseedor
        stmUsuario = con.prepareStatement("UPDATE monederos
SET cantidad = 0 where id_usuario=? and tipo= ?");
        stmUsuario.setString(1, oferta.getId_usuario());
        stmUsuario.setString(2, oferta.getTipo());
        affectedrows = stmUsuario.executeUpdate();

        if (affectedrows > 0) {
        } else {

this.getFachadaAplicacion().muestraExcepcion("Error al borrar al
antiguo poseedor");
            con.rollback();
            return false;
        }
    } else { //diferente
        //Modificar oferta
        stmUsuario = con.prepareStatement("update
hacerofertaventacripto "
            + "set cantidad=cantidad - ? "
            + "where idusuario=? and tipo= ? and
fechaoferta=?");

        stmUsuario.setFloat(1, cantidad);
        stmUsuario.setString(2, oferta.getId_usuario());
        stmUsuario.setString(3, oferta.getTipo());
        stmUsuario.setTimestamp(4, (Timestamp)
oferta.getFechaPublicacion());
        int affectedrows = stmUsuario.executeUpdate();
        if (affectedrows > 0) {
        } else {
            this.getFachadaAplicacion().muestraExcepcion("No
se pudo actualizar la oferta");
            con.rollback();
            return false;
        }
        //modificar poseedor
        stmUsuario = con.prepareStatement("update monederos "
            + "set cantidad = cantidad - ? "
            + "where id_usuario=? and tipo= ?");
        stmUsuario.setFloat(1, cantidad);
        stmUsuario.setString(2, oferta.getId_usuario());
        stmUsuario.setString(3, oferta.getTipo());
        affectedrows = stmUsuario.executeUpdate();
        if (affectedrows > 0) {
        } else {
            this.getFachadaAplicacion().muestraExcepcion("No
se pudo modificar el poseedor vendedor");
            con.rollback();
            return false;
        }
    }
}
//Comprobación comprador xa ten da empresa?

```

5. Desarrollo de funcionalidades

```
consulta = "select * "
          + "from monederos "
          + "where id_usuario=? and tipo=?";
stmCatalogo = con.prepareStatement(consulta);
stmCatalogo.setString(1, u.getId_usuario());
stmCatalogo.setString(2, oferta.getTipo());

rsCatalogo = stmCatalogo.executeQuery();

if (rsCatalogo.next()) {
    //modificar poseedor
    stmUsuario = con.prepareStatement("update monederos "
        + "set cantidad = cantidad + ? "
        + "where id_usuario=? and tipo=?");
    stmUsuario.setFloat(1, cantidad);
    stmUsuario.setString(2, u.getId_usuario());
    stmUsuario.setString(3, oferta.getTipo());
    int affectedrows = stmUsuario.executeUpdate();
    if (affectedrows > 0) {
    } else {
        this.getFachadaAplicacion().muestraExcepcion("No
se pudo modificar al poseedor comprador");
        con.rollback();
        return false;
    }
} else { //aún no ten
    //insertar poseedor
    stmUsuario = con.prepareStatement("insert into
monederos "
        + "values (?, ?, ?)");
    stmUsuario.setString(1, u.getId_usuario());
    stmUsuario.setString(2, oferta.getTipo());
    stmUsuario.setFloat(3, cantidad);
    int affectedrows = stmUsuario.executeUpdate();
    if (affectedrows > 0) {
    } else {
        this.getFachadaAplicacion().muestraExcepcion("No
se pudo insertar al nuevo poseedor");
        con.rollback();
        return false;
    }
}

//Intercambio de fondos
//Vendedor
System.out.println("llegue2");
stmUsuario = con.prepareStatement("update usuarios "
    //Cantidad
    + "set fondos = fondos + (select precio*? from
tablaactualizacionespacios where nombrecripto = ?) "
    + "where id_usuario=? ");
stmUsuario.setFloat(1, cantidad);
stmUsuario.setString(2, oferta.getTipo());
stmUsuario.setString(3, oferta.getId_usuario());

int affectedrows = stmUsuario.executeUpdate();
if (affectedrows > 0) {
} else {
    this.getFachadaAplicacion().muestraExcepcion("No se
pudo modificar fondos vendedor");
    con.rollback();
}
```

```

        return false;
    }
    //Comprador
    stmUsuario = con.prepareStatement("update usuarios "
        + "set fondos = fondos - (select precio*? from
tablaactualizacionespacios where nombrecripto = ?) "
        + "where id_usuario=? ");
    stmUsuario.setFloat(1, cantidad);
    stmUsuario.setString(2, oferta.getTipo());
    stmUsuario.setString(3, u.getId_usuario());

    affectedrows = stmUsuario.executeUpdate();
    if (affectedrows > 0) {
    } else {
        this.getFachadaAplicacion().muestraExcepcion("No se
pudo modificar fondos comprador");
        con.rollback();
        return false;
    }

    stmUsuario = con.prepareStatement("insert into
compraventacriptos "
        + "values (?, ?, ?, CURRENT_TIMESTAMP, ?, ?)");
    stmUsuario.setString(2, oferta.getId_usuario());
    stmUsuario.setString(1, u.getId_usuario());
    stmUsuario.setString(3, oferta.getTipo());
    stmUsuario.setFloat(4, oferta.getCantidad());
    stmUsuario.setFloat(5, oferta.getPrecio());
    affectedrows = stmUsuario.executeUpdate();
    if (affectedrows > 0) {
    } else {
        this.getFachadaAplicacion().muestraExcepcion("Error
insertando tupla compraventa");
        con.rollback();
        return false;
    }
} catch (SQLException e) {
    System.out.println(e.getMessage());
    switch (e.getErrorCode()) {
        case 23502:

this.getFachadaAplicacion().muestraExcepcion("Revisa aquellos campos
que deban ser no nulos.\n");
            break;
        case 23505:
            this.getFachadaAplicacion().muestraExcepcion("El
id_usuario introducido ya se encuentra en uso.\n");
            break;
        default:

this.getFachadaAplicacion().muestraExcepcion(e.getMessage());
            break;
    }
    try {
        if (con != null) {
            con.rollback();
        }
    } catch (SQLException er) {
        System.out.println(er.getMessage());
    }
    return false;
}

```


5. Desarrollo de funcionalidades

```
    } finally {
        try {
            con.commit();
            stmCatalogo.close();
        } catch (SQLException e) {
            System.out.println("Imposible cerrar cursores");
        }
    }
    return resultado;
}
}
```

[F_EXTRA_02]: Encriptación de contraseñas: Se ha optado por la implementación de una función hash que transforme un string en otro string en lugar de un método de encriptación estándar porque, a diferencia de una encriptación que siempre será reversible, la función hash nunca es biyectiva. De esta forma, esta transacción de encriptar contraseñas no es invertible, y en la base de datos nunca se almacenará la contraseña original. Tampoco se podrá acceder, sabiendo el valor de la función hash, a la contraseña inicial.

La contraseña se procesa por cada 2 bytes, siendo cada segmento considerado un long. A continuación, se suman los valores en ascii de cada segmento y se convierte el resultado a un string modificando esta cadena según (1).

```
public int hashPWD(String pwd) {
    long suma = 0, producto = 1, valor = 0;
    for (int i = 0; i < pwd.length(); i++) {
        producto = (i % 2 == 0) ? 1 : producto * 256;
        suma += pwd.charAt(i) * producto;
    }
    return (int) (Math.abs(suma));
}
```

//Obtener String a partir de hash

```
public String obtenerStringHash(int hash, String pwd) {
    String contrasenaEncriptada = "";
    Integer entero = new Integer(hash);
    Character primerCaracterString;

    //(1): Añadir como cabecera un carácter encontrado en una posición
    hash % 10 del string de la contraseña original.
```

```

if (hash % 10 >= 1 && (pwd.length() - hash%10) > 0) {
    primerCaracterString = new Character(pwd.charAt(hash %
10));
    contrasenaEncriptada = primerCaracterString.toString();
}
//En el caso de contraseña demasiado corta:
else
    contrasenaEncriptada = "a3f";
contrasenaEncriptada=contrasenaEncriptada+entero.toString();
return contrasenaEncriptada;
}

```

[F_EXTRA_03] Lectura no comprometida: Se ha desarrollado la lectura no comprometida sobre las solicitudes de dar de alta en las cuentas de usuario, mostradas en la ventana de administrador. Para ello, en lugar de implementar una consulta con SET ISOLATION LEVEL READ UNCOMMITTED, se ha desarrollado la lectura no comprometida a través de establecer un retraso en la transacción.

Así, se puede ejemplificar una lectura sucia si se inician dos sesiones en la aplicación desde el regulador y se ejecuta darDeAlta() sobre la misma tupla, con diferencia de dos segundos en una sesión respecto a otra. El resultado será una excepción SQL con el mensaje 'Imposible cerrar cursores'.

Sin embargo, esta transacción no sirve como ejemplo de ejecución genérica para esta aplicación, pues en primer lugar solo existe un regulador y no habrá dos sesiones simultáneas, y además el retardo en la transacción se ha implementado únicamente para permitir que se obtenga la lectura sucia. Por tanto, en un caso real fuera de este ejemplo no se darán estados inconsistentes.

```

public void darDeAlta(Usuario usuario) {
    Connection con;
    PreparedStatement stmUsuario = null;
    con = super.getConexion();
    try {
        stmUsuario = con.prepareStatement("UPDATE usuarios SET enespera =
0 WHERE id_usuario = ?; ");
        usuario.setEnEspera(0);
        stmUsuario.setString(1, usuario.getId_usuario());
        stmUsuario.executeUpdate();
        TimeUnit.SECONDS.sleep(10);
    } try {
}

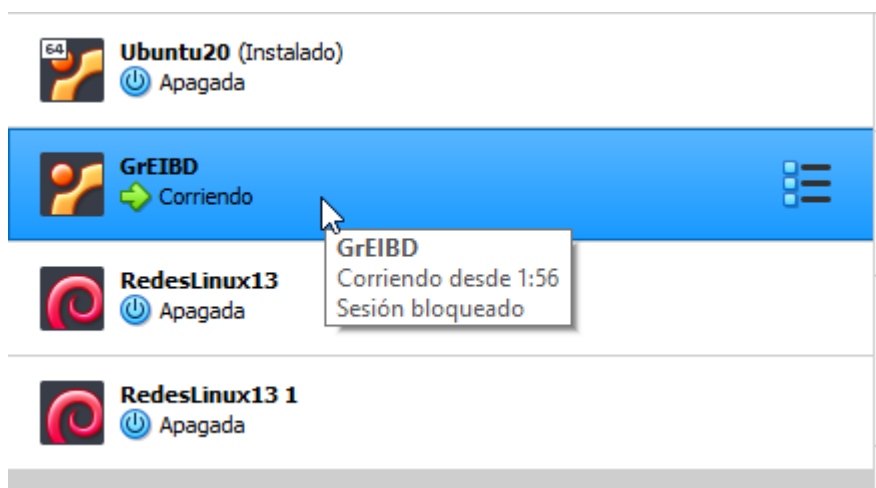
```

5. Desarrollo de funcionalidades

```
catch (InterruptedException e) {
System.err.format("IOException: %s%n", e);
}
catch (SQLException e) {
System.out.println(e.getMessage());
} finally {
try {
con.commit();
stmUsuario.close();
} catch (SQLException e) {
System.out.println("Imposible cerrar cursores");
}
}
```

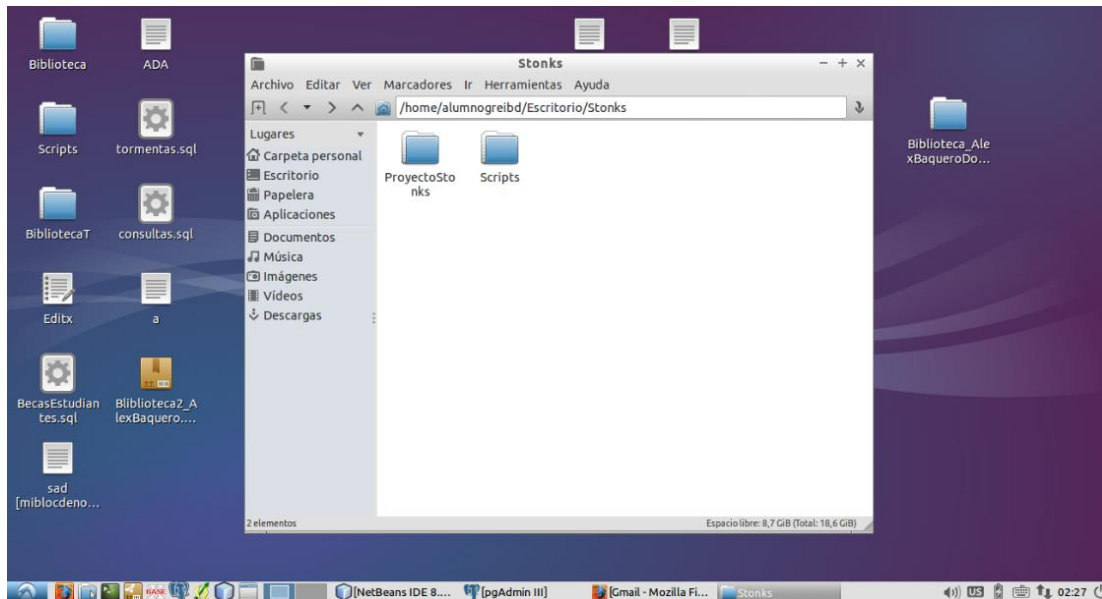
6. Apéndice

En este apartado se mostrará a poner en marcha la aplicación “Stonks” en la máquina virtual de la materia.

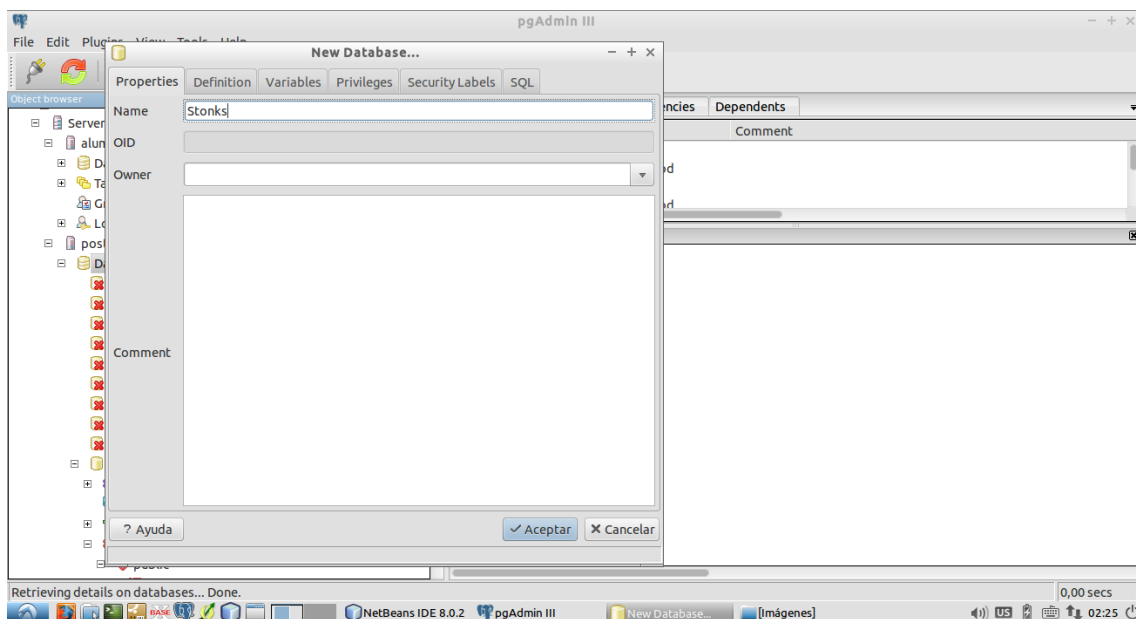


Una vez seleccionado el Oracle VM VirtualBox y la máquina virtual GrEIBD (máquina virtual proporcionada en el campus virtual de la asignatura Bases de Datos II), la información suministrada comenzará con la creación de la base de datos y código del proyecto.

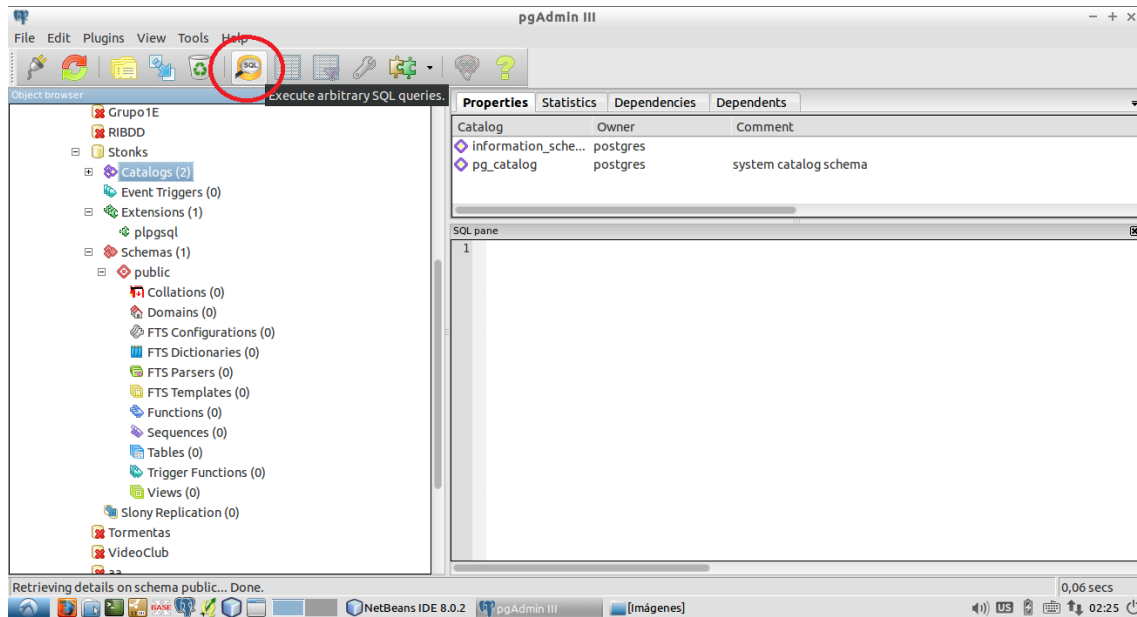
6. Apéndice



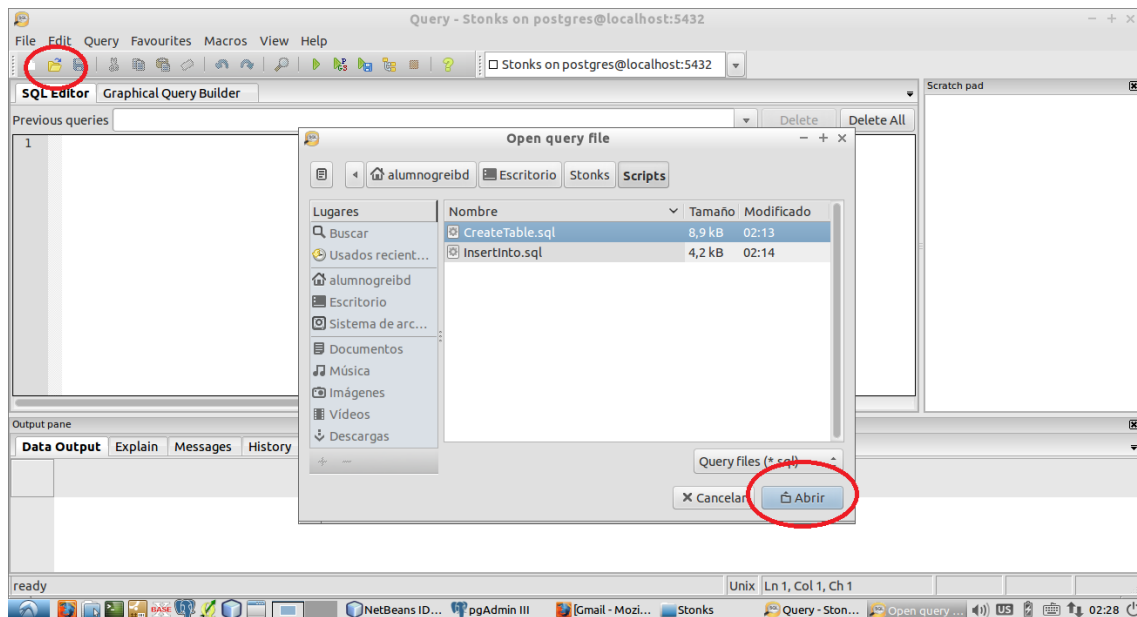
En primer lugar, abrimos el pgAdmin III, desplegamos ServerGroups, Server y un localhost hasta quedarnos en Databases. Seleccionamos click derecho y creamos una base de datos nueva clickeando “New Database...”. Introducimos el nombre de nuestro sistema (en nuestro caso, Stonks”) y aceptamos.



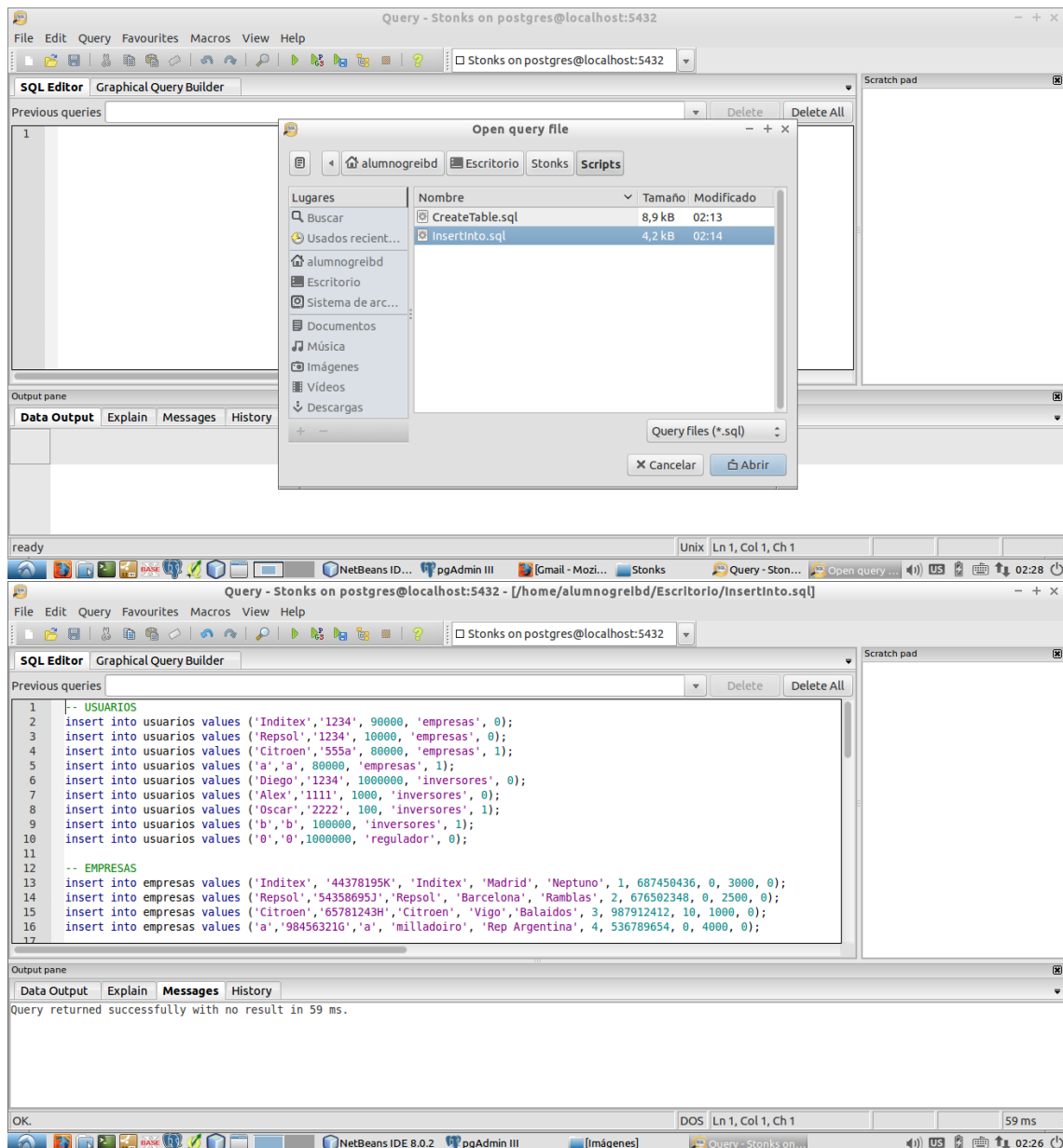
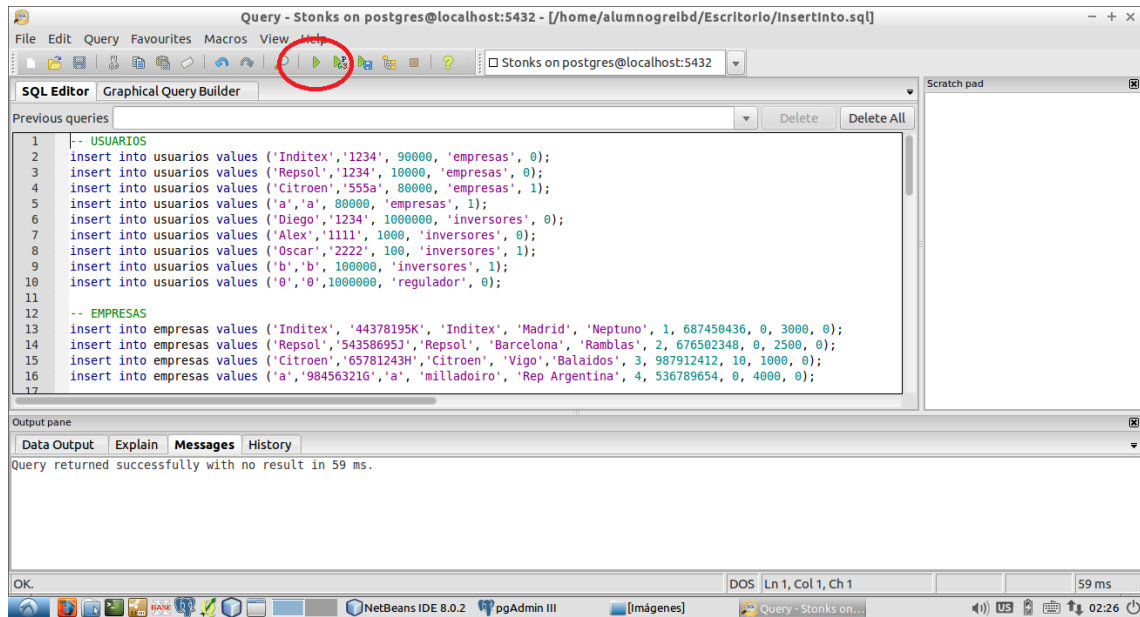
Una vez creada la base de datos, escogemos el icono de la lupa de SQL.



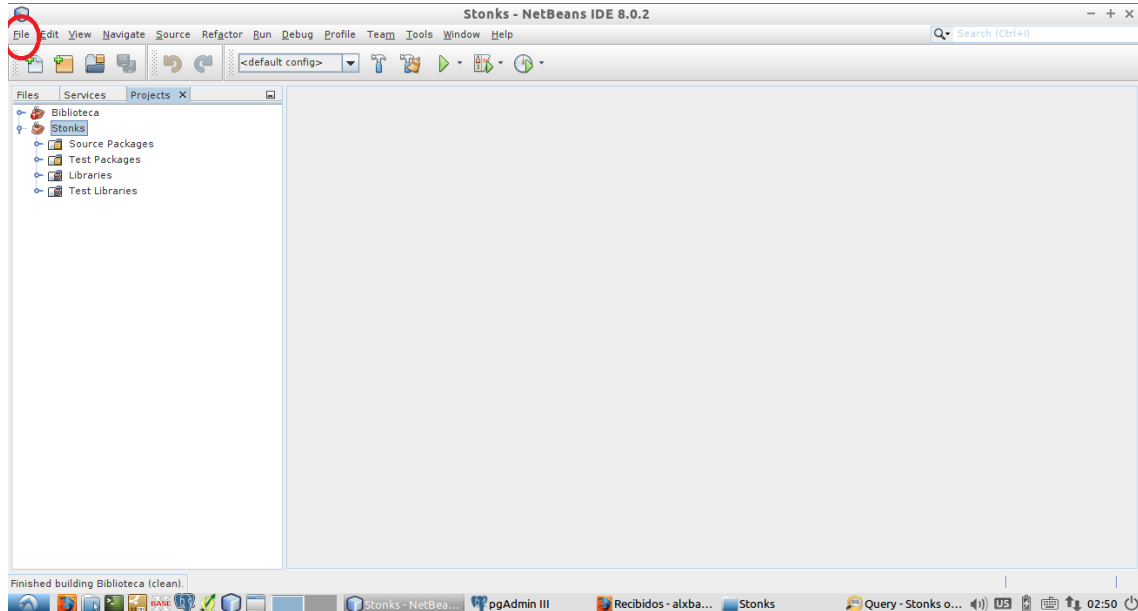
Seleccionamos el icono de la carpeta amarilla, escogemos los documentos .sql: CreateTable y Insertinto y los abrimos por separado. Por último, le damos al triángulo verde para ejecutar.



6. Apéndice

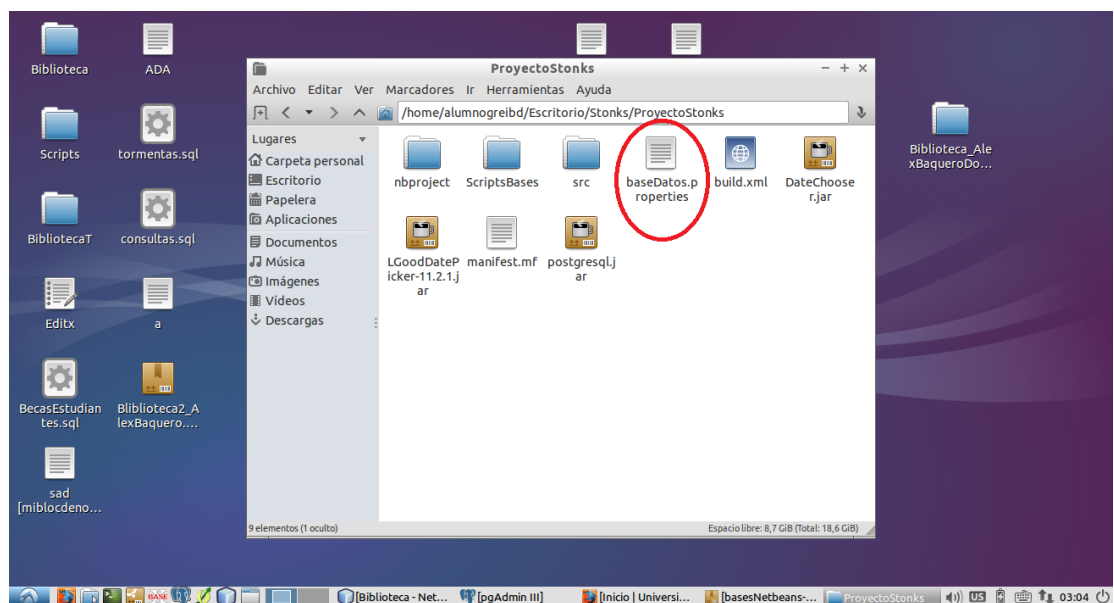


En segundo lugar, abrimos Netbeans, clickeamos en File, “Abrir Proyecto” y seleccionamos el proyecto Stonks.



Por último, mencionar acerca de la información de acceso a la base de datos y los nombres de usuario y contraseñas necesarias tanto en la base de datos como en la aplicación.

- Todas las bases tienen un puerto de defecto que es 5432.
- En BasesDatos se tiene que poner el nombre de la base creada en el PgAdmin III.
- Todos los usuarios de bases de datos tienen un nombre de usuario y una contraseña. En usuario y clave se debe colocar esto estas características



6. Apéndice

