

# AGENCIA DE CASTINGS

Rodrigo Roca Román

Javier Alonso Lledó

Grado en Ingeniería Informática

12:00-14:00

29-10-2020

## Contenido

Fase 1 .....	3
Diccionario de datos.....	3
Modelo Entidad-Relación Extendido.....	6
Fase 2 .....	7
Transformación del modelo ER al modelo relacional .....	7
Modelo relacional en Pgmodeler .....	8
Consultas SQL.....	9
Fase 3 .....	16
Normalización .....	16
Creación de roles y usuarios .....	16
Implementación de disparadores .....	18
Acceso desde Java .....	26

## Fase 1

### Diccionario de datos

Entidad	Atributo	Dominio	Restricción(Pk, FK,...)
CLIENTE	codigo_cliente	char[10]	PK, *
CLIENTE	nombre	char[40]	***
CLIENTE	direccion	dirección	**
CLIENTE	telefono	int[9]	
CLIENTE	persona_de_contacto	char[40]	*
CLIENTE	actividad	char[1]	{M,P}(P= Publicidad y cine , M= moda)
CASTING	codigo_casting	char[10]	PK, *
CASTING	nombre	char[40]	***
CASTING	descripcion	char[1000]	
CASTING	fecha_de_contratacion	date	
CASTING	coste	int	>0
ONLINE	numero_de_personas	int	>0
ONLINE	fecha	date	
ONLINE	plataforma_web	URL	Debe empezar por https://www y terminar en .com o .es .
PRESENCIAL	numero_de_personas	int	>0
AGENTE	DNI	char[9]	****
AGENTE	nombre	char[40]	***
AGENTE	direccion	dirección	**
FASE	codigo_fase	int	PK,>0
FASE	fecha_de_inicio	date	
PRUEBA INIVIDUAL	numero	int	PK,>0
PRUEBA INIVIDUAL	fecha	date	
PRUEBA INIVIDUAL	sala_de_celebracion	dirección	**
PRUEBA INIVIDUAL	descripcion	char[1000]	
PRUEBA INIVIDUAL	coste	int	>0
CANDIDATOS	codigo_candidato	char[10]	PK, *
CANDIDATOS	nombre	char[40]	***
CANDIDATOS	direccion	dirección	**
CANDIDATOS	telefono	int[9]	
CANDIDATOS	fecha_de_nacimiento	date	
CANDIDATOS	resultado_prueba	char[1]	{Y,N},(Y=Aprobado ,N=Suspenso)
CANDIDATOS	importe_total	int	

REPRESENTANTE	NIF	char[9]	PK, ****
REPRESENTANTE	nombre	char[40]	***
REPRESENTANTE	telefono	int[8]	
REPRESENTANTE	direccion	dirección	
PERFIL	codigo_de_perfil	char[10]	PK, *
PERFIL	provincia	provincia	
PERFIL	sexo	char[1]	{M,F}(M=Masculino, F= Femenino)
PERFIL	altura	Int	>0
PERFIL	edad	Int	>0
PERFIL	color_del_pelo	char[20]	
PERFIL	color_de_ojos	char[20]	
PERFIL	especialidad	char[1]	{M,A}(M=modelo, A= Actor)
PERFIL	experiencia	char[1]	{Y,N}(Y= Si , N= No)
ADULTO	DNI	char[9]	PK, ****
NIÑO	nombre_tutor	char[40]	PK, ***
contrata	codigo_casting	char[10]	PK, *
contrata	codigo_candidato	char[10]	PK, *

\*→ Un código contiene tanto números como letras pudiendo tener una longitud fija de 10 caracteres.

\*\*→ Una dirección contiene el siguiente formato: Nº/ Calle

\*\*\*→ Solo letras y espacio entre palabras, primera de cada palabra en mayúsculas

\*\*\*\*→ <http://www.interior.gob.es/web/servicios-al-ciudadano/dni/calculo-del-digito-de-control-del-nif-nie> .

Los 8 primeros numéricos, el último alfabético, resto de dividir el número por 23

Relación	Entidades	Cardinalidad	Atributos	Dominio
Contrata	CLIENTE CASTING	(0,1) (0,n)	coste	int
Es	CASTING(MADRE) ONLINE(HIJA) PRESENCIAL(HIJA)	(1,1) (0,1) (0,1)		
Dirige	PRESENCIAL AGENTE	(1,1) (1,1)		
Necesita	CASTING PERFIL	(1,n) (1,n)		
Consta De	PRESENCIAL FASE	(1,1) (1,n)		
Se divide en	FASE PRUEBA INDIVIDUAL	(1,1) (1,n)		
Realiza	PRUEBA INDIVIDUAL CANDIDATO	(0,n) (1,n)	resultado_prueba	char[1]{Y,N} (Y=Aprobado, N=)
Corresponde	CANDIDATO	(0,1)		

	PERFIL	(1,1)		
Gestiona	CANDIDATO REPRESENTANTE	(1,n) (0,1)		
Es	PERFIL(MADRE) ADULTO(HIJA) NIÑO(HIJA)	(1,1) (0,1) (0,1)		

#### Documentación detallada de entidades y relaciones

La entidad **CLIENTE** representa a un cliente de la compañía de catering con sus datos básicos como un **nombre, dirección, teléfono, etc...** . El **CLIENTE** al utilizar la relación **Contrata** debe proporcionar al **CASTING** el **coste** de producción.

El **CASTING** tiene que ser o **PRESENCIAL** u **ONLINE** y se encarga mediante la relación **Se corresponde** de mostrar todos los perfiles (**PERFIL**) de los candidatos(**CANDIDATO**) para que sean seleccionados por el **CLIENTE** ,permitir que los **CANDIDATOS** usen la relación **realiza** para participar las pruebas individuales (**PRUEBA INDIVIDUAL**) al ser seleccionados por un **CASTING**.

Los castings presenciales (**CASTING PRESENCIAL**) son dirigidos (relación **Dirige**) por un **AGENTE** y **consta de** una o varias fases (**FASE**).

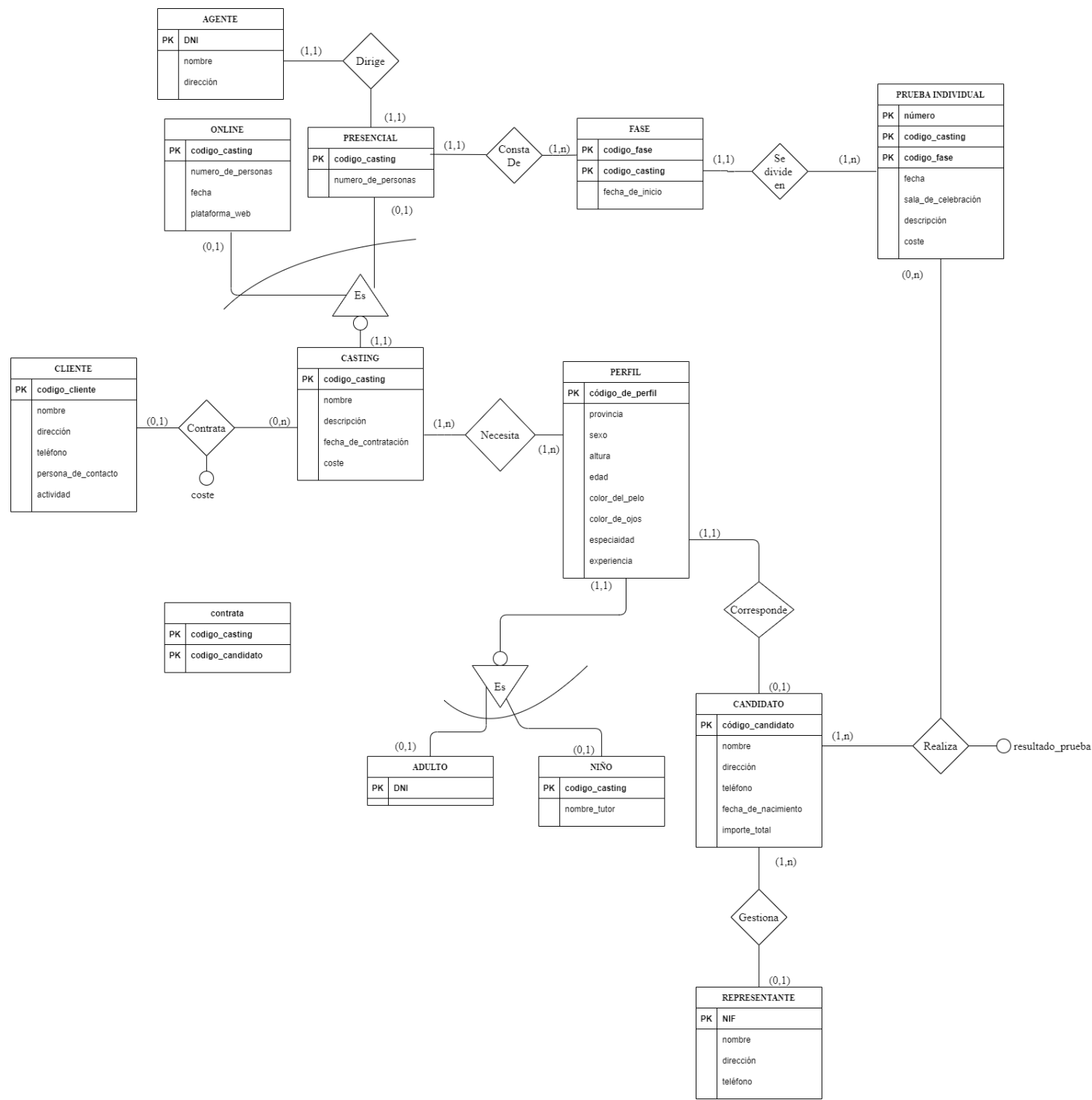
Las fases (**FASE**) de los castings presenciales (**CASTING PRESENCIAL**) están numeradas y ordenadas dependiendo del número de fases que tenga el casting. Cada **FASE** se divide en una o varias pruebas individuales (**PRUEBA INDIVIDUAL**) con el objetivo de calificar a los diversos candidatos de una audición. Para ello se utiliza la relación **Realiza** y devuelve al **CANDIDATO** el **resultado\_prueba** que puede ser positivo o negativo indicando que el **CANDIDATO** ha superado o no la prueba.

Cada **CANDIDATO** puede tener o no un **REPRESENTANTE** que **Gestiona** las acciones y datos de este. Además a cada **CANDIDATO** le **Corresponde** un único **PERFIL** con todas las características físicas y laborales de este.

Los perfiles (**PERFIL**) tienen que ser o un **ADULTO** o de **NIÑO** cada uno con sus datos únicos y con los que en total serán utilizados para poder participar en un **CASTING**.

La tabla **contrata** tiene como objetivo guardar a todos los **CANDIDATOS** que han sido contratados para los diversos **CASTINGS**

# Modelo Entidad-Relación Extendido



## Fase 2

### Transformación del modelo ER al modelo relacional

Cliente (codigo\_cliente, nombre, direccion, telefono, persona\_de\_contacto, actividad)

Casting (codigo\_casting, nombre, descripcion, fecha\_de\_contratacion, **codigo\_cliente**, coste)

Online (**codigo\_casting**, numero\_de\_personas, fecha, plataforma\_web)

Presencial (numero\_de\_personas, **codigo\_casting**, DNI)

Agente (DNI, nombre, direccion)

Fase (codigo\_fase, fecha\_de\_inicio, **codigo\_casting**)

Prueba individual (numero, fecha, sala\_de\_celebración, descripcion, coste, **codigo\_fase**, **codigo\_casting**)

Candidatos (codigo\_candidato, nombre, direccion, telefono, fecha\_de\_nacimiento, importe\_total, **nif\_representante**, **codigo\_de\_perfil**)

Niño (nombre\_tutor, **codigo\_candidato**)

Adulto (DNI, **codigo\_candidato**)

Representante (NIF, nombre, telefono, direccion)

Perfil (codigo\_de\_perfil, provincia, sexo, altura, edad, color\_del\_pelo, color\_de\_ojos, especialidad, experiencia)

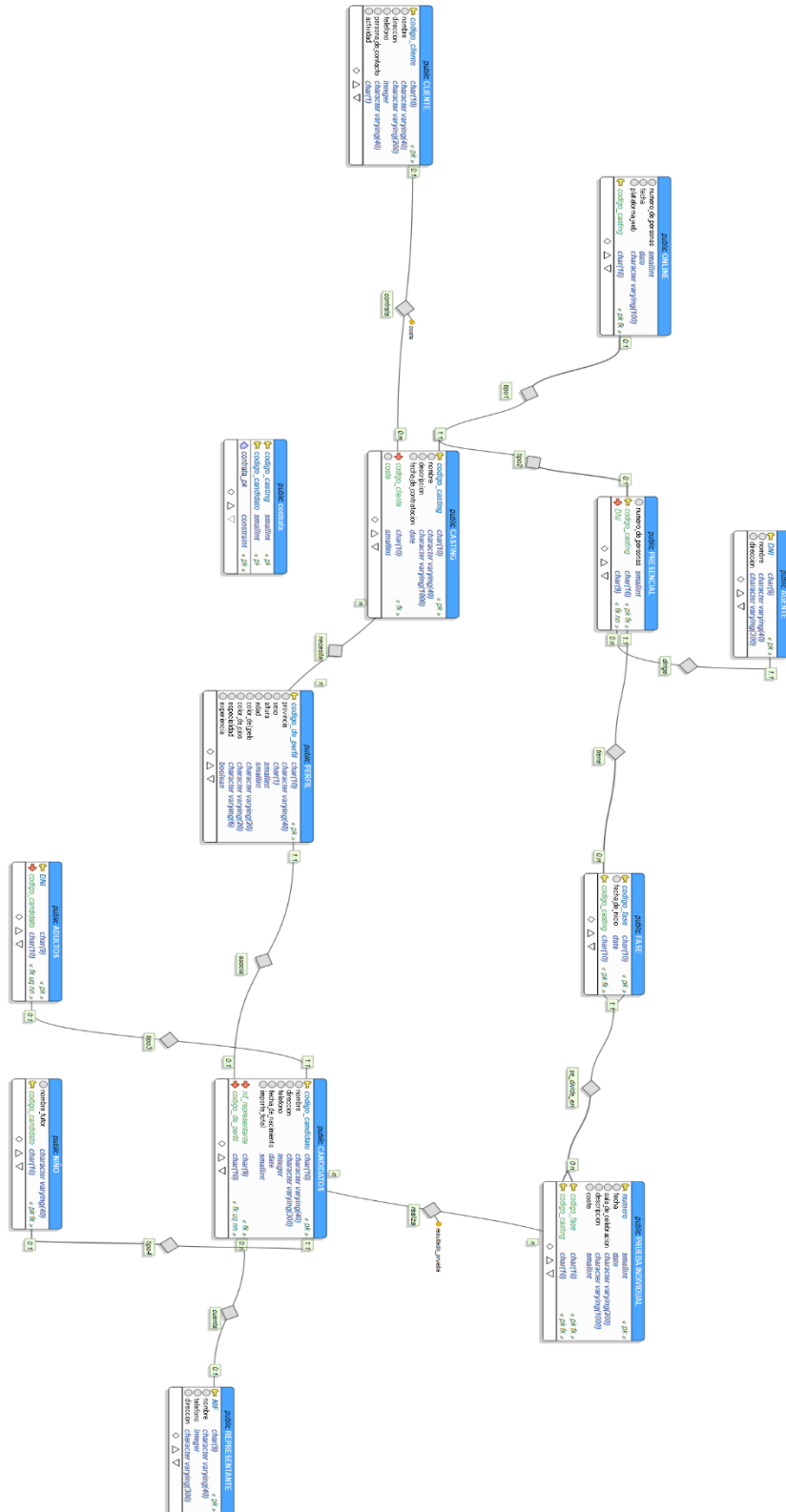
Casting\_necesita\_perfil (**codigo\_casting**, **codigo\_de\_perfil**)

Candidato\_realiza\_prueba (**codigo\_candidato**, **numero**, **codigo\_fase**, **codigo\_casting**, resultado\_prueba)

contrata(**codigo\_casting**, **codigo\_candidato**)

## Modelo relacional en Pgmodeler

Se incluirá una captura de pantalla del diagrama generado. Se incluirán en la entrega el fichero .dbm obtenido y el fichero .sql generado.





## Consultas SQL

1) Mostrar el nombre de los clientes que no han contratado ningún casting.

-- 1

```
select nombre from cliente where codigo_cliente not in  
(select codigo_cliente from casting);
```

	nombre character varying (40)
1	Jesus
2	Antonio
3	Luis
4	Andrea
5	Vicente
6	Helena
7	Álvaro

2) Mostrar el nombre de los candidatos que han superado todas las pruebas.

-- 2

```
select nombre from candidatos where codigo_candidato not in  
(select codigo_candidato from candidato_realiza_prueba where resultado_prueba = false);
```

Data Output	Explain	Message
	nombre character varying (40)	
1	Antonio	
2	Alba	
3	Clara	
4	Fernando	
5	Antonio	
6	Miguel	

### 3) Mostrar el número de candidatos que hay asociados a cada perfil.

--3

```
select codigo_de_perfil, count(codigo_de_perfil) as num_candidatos from candidatos
group by codigo_de_perfil;
```

	codigo_de_perfil character (10)	num_candidatos bigint
1	F1SHW0I1FX	1
2	4FH80JER99	1
3	ZFMIOI42PS	1
4	QAU8GWQRXD	1
5	J4R9OQEUW3	1
6	5R71U58U1Z	1
7	DD1IYABIES	1
8	29T06C7ALK	1
9	KPW6TUZKDQ	1
10	55GH5Y203E	1

### 4) Mostrar el nombre del empleado que más castings ha dirigido.

--4

```
select nombre from agente where dni in
(select dni from presencial group by dni having count(dni) =
(select max(num_dni) from (select dni, count(dni) as num_dni from presencial group by dni) as cuantosdni));
```

	nombre character varying (40)
1	Cain

### 5) Mostrar el número de candidatos que se han presentado a cada casting (que al menos hayan realizado una prueba).

--5

```
select casting.codigo_casting, count(distinct codigo_candidato) from casting
inner join candidato_realiza_prueba on
casting.codigo_casting = candidato_realiza_prueba.codigo_casting
group by casting.codigo_casting;
```

	codigo_casting [PK] character (10)	count bigint
1	9C2JA0T52E	1
2	9ETDJDGZOC	1
3	HK328Z8E63	1
4	I96ICE30U7	1
5	ILBNLLC6TW	1
6	QF8BYWQFU	1
7	RQ3TCC0MTS	1
8	ZDI2J077V0	1
9	ZQ8IT7ABC7	1

6) Mostrar el nombre y la dirección de las candidatas que tengan el pelo rubio y sean de Madrid.

```
--6
select nombre , direccion from candidatos where
codigo_de_perfil in (select codigo_de_perfil from perfil where color_del_pelo ='Rubio' and provincia = 'Madrid');
```

	nombre character varying (40)	direccion character varying (300)
1	Isabel	3616 Tremblay Creek
2	Aurelio	52948 Hodkiewicz Key

7) Mostrar el código de perfil de los perfiles requeridos en los castings que incluyen la subcadena “anuncio tv” en su descripción.

```
--7
select codigo_de_perfil from casting_necesita_perfil
where codigo_casting in
(select codigo_casting from casting where descripcion like '%anuncio tv%');
```

	codigo_de_perfil character (10)
1	4FH8OJER99
2	QAU8GWQRXD

8) Mostrar el código de los candidatos que tienen representante y tienen el pelo castaño.

```
--8
select codigo_candidato from candidatos where codigo_de_perfil in (select codigo_de_perfil from perfil where color_del_pelo ='Castaño')
and nif_representante != 'null';
```

	codigo_candidato [PK] character (10)
1	SJ1Q6TM842
2	I2ZPQSTI3X

9) Mostrar el precio total que ha de pagar cada candidato.

--9

```
select codigo_candidato, sum(coste) as dinero_a_pagar
from candidato_realiza_prueba inner join prueba_individual
on prueba_individual.numero = candidato_realiza_prueba.numero and
prueba_individual.codigo_fase = candidato_realiza_prueba.codigo_fase and
prueba_individual.codigo_casting = candidato_realiza_prueba.codigo_casting
group by codigo_candidato;
```

	codigo_candidato character (10)	dinero_a_pagar bigint
1	770C6RCUSM	250
2	I2ZPQSTI3X	77
3	0GMYVXM68R	267
4	NK9IRY891Y	55
5	EWXLXMLBEW	413
6	4QU32DOAEZ	207
7	TZCCRGP35Y	885

10) Mostrar el número de candidatos adultos y el número de candidatos niños que hay en la base de datos.

--10

```
select count(*) as adultos, (select count(*) from niño) as niño from adultos;
```

	adultos bigint	niño bigint
1	8	2

11) Mostrar el dni del agente que ha dirigido el casting en el que alguna prueba individual se ha llevado a cabo en la sala "flor".

```
--11
select dni from presencial where codigo_casting in
(select codigo_casting from prueba_individual where sala_de_celebracion = 'Sala flor');
```

	dni character (9)
1	36002833J
2	84511807S
3	42886401A

12) Mostrar la plataforma web que se ha usado en el casting online más caro, así como el nombre del cliente que ha contratado dicho casting.

```
--12
select plataforma_web , cliente.nombre from casting inner join cliente on casting.codigo_cliente = cliente.codigo_cliente
inner join online on casting.codigo_casting = online.codigo_casting
where coste = (select max(mayor_coste) from (select coste as mayor_coste from casting) as costes);
```

	dni character (9)
1	36002833J
2	84511807S
3	42886401A

13) Mostrar el porcentaje de clientes que hay de cada tipo

```
--13
select actividad, count(actividad)*100/(select count(*) from cliente) as porcentaje
from cliente group by actividad;
```

	actividad character (1)	porcentaje bigint
1	P	50
2	M	50

14) Mostrar el nombre de los candidatos que han superado alguna prueba de algún casting, así como el nombre del casting.

```
--14
select distinct candidatos.nombre, casting.nombre from candidato_realiza_prueba inner join candidatos
on candidato_realiza_prueba.codigo_candidato = candidatos.codigo_candidato
inner join casting on candidato_realiza_prueba.codigo_casting = casting.codigo_casting
where resultado_prueba = true ;
```

	nombre character varying (40)	nombre character varying (40)
1	Andrea	VOCALISTA PARA METALICA
2	Antonio	BUSCAMOS MODELOS DE B...
3	Alba	MISS FANTASTICO
4	Andrea	NATURALISTAS FLORES
5	Isabel	SESIÓN DE FOTOS
6	Miguel	NIÑOS PARA LOS GOONIES

15) Mostrar el dinero total recaudado por la empresa.

```
--15
select sum(coste) from casting;
```

	sum bigint
1	88003

16) Mostrar el nombre y el teléfono de los representantes que representen a 2 candidatos como mínimo.

```
--16
select nombre , telefono from representante where nif in (select nif_representante from candidatos
group by nif_representante having count(nif_representante)>=2);
```

	nombre character varying (40)	telefono integer
1	Alberto	674761306
2	Juan	674761306

17) Mostrar el dni de los adultos que no tengan representante

```
--17
select dni from candidatos inner join adultos
on candidatos.codigo_candidato = adultos.codigo_candidato
where nif_representante is null;
```

	dni [PK] character (9)
1	39790452S
2	62598751L
3	49671721R

18) Mostrar los datos del perfil más demandado así como el nombre del cliente que lo ha requerido para su casting.

```
--18
select cliente.nombre as nombre_cliente, perfil.* from casting_necesita_perfil inner join casting on
casting_necesita_perfil.codigo_casting = casting.codigo_casting
inner join perfil on
casting_necesita_perfil.codigo_de_perfil = perfil.codigo_de_perfil
inner join cliente on
casting.codigo_cliente = cliente.codigo_cliente
where perfil.codigo_de_perfil in
(select codigo_de_perfil from casting_necesita_perfil group by codigo_de_perfil
having count(codigo_de_perfil) =
(select max(veces_solicitado) from (select codigo_de_perfil, count(codigo_de_perfil) as veces_solicitado
from casting_necesita_perfil
group by codigo_de_perfil) as cuantosperfil));
```

	nombre_cliente character varying (40)	codigo_de_perfil character (10)	provincia character varying (20)	sexo character (1)	altura smallint	edad smallint	color_del_pelo character varying (20)	color_de_ojos character varying (20)	especialidad character varying (6)	experiencia boolean
1	Javier	4FH80JER99	Girona	M	184	25	Castaño	Castaños	Actor	true
2	Jorge	4FH80JER99	Girona	M	184	25	Castaño	Castaños	Actor	true
3	Carlos	4FH80JER99	Girona	M	184	25	Castaño	Castaños	Actor	true

19) Mostrar el número de pruebas superadas por cada niño.

```
--19
select count(*), niño.codigo_candidato from candidato_realiza_prueba inner join candidatos
on candidato_realiza_prueba.codigo_candidato = candidatos.codigo_candidato
inner join niño on niño.codigo_candidato = candidatos.codigo_candidato
where resultado_prueba = 'true' group by niño.codigo_candidato;
```

	count bigint	codigo_candidato [PK] character (10)
1	1	I2ZPQSTI3X

20) Mostrar el código del casting que más fases tiene.

```
--20
select codigo_casting from fase
group by codigo_casting having count(codigo_casting) =
(select max(nfases) from
(select codigo_casting, count(codigo_casting) as nfases from fase
group by codigo_casting)
as cuantasfases);
```

## Fase 3

### Normalización

Análisis de dependencias funcionales y normalización de cada tabla debidamente justificada.

Está en 1FN, dominios atómicos y no hay atributos multivaluados.

Está en 2FN, todo atributo depende totalmente de las PKs.

Está en 3FN, no hay DF transitivas.

Ya está normalizado.

### Creación de roles y usuarios

Para poder crear los roles administrador , gestor y recepcionista hemos utilizado la siguiente operación: **create role “nombre del rol a crear”;**

Después de crear los roles utilizamos la operación **grant “lista de privilegios” on “tablas” to “rol”;** para asignar los distintos privilegios sobre las tablas a cada rol.

El código siguiente es el utilizado en la práctica para crear los roles y asignarles los distintos privilegios:

```
create role administrador;
```

```
grant all privileges on adultos, agente, candidato_realiza_prueba, candidatos, casting,  
casting_necesita_perfil, cliente, fase, niño, online, perfil,
```

```
presencial, prueba_individual, representante to administrador;
```

```
create role gestor;
```

```
grant select, update, delete, insert on adultos, agente, candidato_realiza_prueba, candidatos,  
casting, casting_necesita_perfil, cliente, fase, niño, online, perfil,
```

```
presencial, prueba_individual, representante to gestor;
```

```
create role recepcionista;
```

```
grant select on adultos, agente, candidato_realiza_prueba, candidatos, casting,  
casting_necesita_perfil, cliente, fase, niño, online, perfil,
```

```
presencial, prueba_individual, representante to recepcionista;
```

Con los roles ya creados solo falta crear y configurar los usuarios. Para ello utilizamos la operación **create user “nombre de usuario” password ‘contraseña’;** para crearlos y la operación **grant “rol” to “usuario”;** para asignarle el rol que corresponda a cada usuario.

```
create user javier password 'javier';
```

```
create user rodrigo password 'rodrigo';
```

```
create user jefe password 'jefe';
```



*grant administrador to jefe;*

*grant gestor to rodrigo;*

*grant recepcionista to javier;*

En el proyecto se encontrarán dos ficheros donde se pueda obtener este código siendo el primero el fichero *RolesUser.txt* y el segundo *SetUpBD-PL3*.

Los usuarios con sus contraseñas creados para esta práctica son los siguientes:

- Usuario: javier , Rol: Recepcionista , Contraseña: javier
- Usuario: rodrigo , Rol: Gestor , Contraseña: rodrigo
- Usuario: jefe , Rol: Administrador , Contraseña: jefe

Ejemplo de operación válida y operación no válida para el usuario javier con el rol de recepcionista (Privilegios: SELECT).

Operación permitida:

```

1 select * from cliente;
2

```

Data Output	Explain	Notifications	Query History	Messages		
<div> <div></div> <div> <div>codigo_cliente</div> <div>[PK] character (10)</div> </div> <div></div> </div>	<div> <div></div> <div>nombre</div> <div>character varying (40)</div> </div> <div></div>	<div> <div></div> <div>direccion</div> <div>character varying (200)</div> </div> <div></div>	<div> <div></div> <div>telefono</div> <div>integer</div> </div> <div></div>	<div> <div></div> <div>persona_de_contacto</div> <div>character varying (40)</div> </div> <div></div>	<div> <div></div> <div>actividad</div> <div>character (1)</div> </div> <div></div>	
1	G6X585PF10	Javier	26287 Laila Courts	922035546	Maria	P
2	Z0EF2Y4ZTC	Rodrigo	1957 Sporer Place	951429429	Carmen	P
3	ASS4APN67H	Fernando	1957 Sporer Place	982805480	Oscar	P
4	K1DI6DFQK4	Jesus	100 Cartwright Port	620786646	Alex	P

Operación no permitida:

```
2 delete from cliente;
```

Data Output	Explain	Notifications	Query History	Messages
ERROR: permiso denegado a la tabla cliente SQL state: 42501				

Ejemplo de operación válida y operación no válida para el usuario rodrigo con el rol de gestor (Privilegios: SELECT, INSERT, UPDATE, DELETE).

Operación permitida:

```
1 delete from niño;
```

Data Output	Explain	Notifications	Query History	Messages
DELETE 2  Query returned successfully in 55 msec.				

Operación no permitida:

```

1 CREATE TABLE public.persona (
2     nombre_tutor character varying(40),
3     codigo_candidato char(10) NOT NULL,
4     CONSTRAINT persona_pk PRIMARY KEY (codigo_candidato)
5
6 );

```

Data Output Explain Notifications Query History Messages

ERROR: permiso denegado al esquema public

LINE 1: CREATE TABLE public.persona (  
  ^

SQL state: 42501

Character: 14

Ejemplo de operación válida y operación no válida para el usuario rodrigo con el rol de gestor (Privilegios: ALL).

Operación permitida:

```

1 select * from cliente;

```

Data Output Explain Notifications Query History Messages

	codigo_cliente [PK] character (10)	nombre character varying (40)	direccion character varying (200)	telefono integer	persona_de_contacto character varying (40)	actividad character (1)
1	G6X585PF10	Javier	26287 Laila Courts	922035546	Maria	P
2	Z0EF2Y4ZTC	Rodrigo	1957 Sporer Place	951429429	Carmen	P
3	ASS4APN67H	Fernando	1957 Sporer Place	982805480	Oscar	P
4	K1DI6DFQK4	Jesus	100 Cartwright Port	620786646	Alex	P

Operación no permitida:

No hay ninguna operación no permitida para el rol administrador.

Implementación de disparadores

### Primer disparador:

```
-- PRIMER TRIGGER
--Calcular el importe total que ha pagado cada candidato
--teniendo en cuenta todas las pruebas que ha realizado.

create or replace function fun_importe_candidato() returns Trigger as
$actualizarImporte$
begin
    update candidatos
    set importe_total = importe_total + (select coste from prueba_individual where
    numero = new.numero and
    codigo_fase = new.codigo_fase and
    codigo_casting = new.codigo_casting)

    where codigo_candidato = new.codigo_candidato;

    return new;
end
$actualizarImporte$
Language plpgsql;

create trigger tr_importe_candidato after insert on candidato_realiza_prueba
for each row
execute procedure fun_importe_candidato();
```

Cuando se inserta una nueva tupla en la tabla candidato\_realiza\_prueba, la cual contiene toda la información necesaria para saber qué candidato ha hecho qué prueba (y es la manera de indicar que un candidato ha realizado una prueba en nuestra base de datos), se activa este disparador.

Lo que hace este disparador es obtener el coste de la prueba almacenado en la tabla prueba\_individual y añadirlo al total del importe que el candidato debe pagar por realizar las pruebas que realice. El campo que almacena este número es importe\_total y está en la tabla candidatos.

El cómo funciona se fundamenta en hacer un update en la tabla candidatos y actualizando el valor importe\_total sumando el valor que ya había más el de la prueba realizada por el candidato. El valor de importe\_total cuando se inserta un candidato nuevo es, por defecto, 0.

### Comprobación:

```
-- Prueba trigger 1:
-- Hacemos que el candidato 'TZCCRG35Y' haga una prueba cuyo coste es 110
-- Importe candidato (llamado importe_total) antes del trigger:
select importe_total from candidatos where codigo_candidato = 'TZCCRG35Y';
-- Despues del trigger:
insert into candidato_realiza_prueba values ('TZCCRG35Y','02','LMBJHFWLQE','ZDI2J077V0','TRUE');
select importe_total from candidatos where codigo_candidato = 'TZCCRG35Y';
```

### Antes del disparador:

Data Output	Explain	Messages
importe_total smallint		
1	670	

Después del disparador:

Data Output	Explain	Message
importe_total smallint		
1	780	

Segundo disparador:

```
-- SEGUNDO TRIGGER
--Cuando un candidato supera una prueba de un casting, se debe calcular el número de pruebas superadas por dicho candidato.
--Si ese número coincide con el número de pruebas totales de dicho casting, se mostrará un mensaje y se insertará en la tabla "contrata".

create or replace function fun_insertar_contrata() returns Trigger as
$insertarContratados$
declare
    num_pruebas_totales smallint;
    num_pruebas_pasadas smallint;
begin
    if(new.resultado_prueba = 'TRUE') then

        num_pruebas_pasadas = (select count(*) from candidato_realiza_prueba where codigo_candidato = new.codigo_candidato
                                and resultado_prueba = 'TRUE');
        num_pruebas_totales = (select count(*) from prueba_individual where codigo_casting = new.codigo_casting);

        if(num_pruebas_pasadas = num_pruebas_totales) then
            raise notice 'El candidato ha superado todas las pruebas, está contratado';
            insert into contrata values (new.codigo_casting , new.codigo_candidato);
        end if;
    end if;
    return new;
end
$insertarContratados$
Language plpgsql;

create trigger tr_insertar_contrata after insert on candidato_realiza_prueba
for each row
execute procedure fun_insertar_contrata();
```

El evento que hace que este disparador se active es el mismo que en el caso anterior: una inserción en la tabla candidato\_realiza\_prueba.

Lo que hace este disparador es que si un candidato ha superado todas las pruebas (el campo resultado\_prueba tiene que ser TRUE en todas las pruebas) avisa con un raise notice de que se han superado todas las pruebas e inserta los valores del candidato que ha realizado la prueba y del casting en el que participa en la tabla *contrata*.

Si el resultado de la prueba que ha hecho que el disparador se active es FALSE, entonces el disparador no hace nada ya que es evidente que no ha superado todas las pruebas (y así se ahorra tiempo).

### Comprobación:

```
-- Prueba trigger 2:
-- Vamos a hacer que un candidato supere todas las pruebas para comprobar el funcionamiento
-- (necesita haber hecho la prueba que se inserta en la comprobación del trigger 1):
-- insert into candidato_realiza_prueba values ('TZCCRGP35Y','02','LMBJHFWLQE','ZDI2J077V0','TRUE');
insert into candidato_realiza_prueba values ('TZCCRGP35Y','03','LMBJHFWLQE','ZDI2J077V0','TRUE');
select * from contrata where codigo_candidato = 'TZCCRGP35Y';
```

Al hacer el insert se activa el disparador:

Data Output	Explain	Messages	Notifications
NOTICE: El candidato ha superado todas las pruebas, está contratado			
INSERT 0 1			
Query returned successfully in 40 msec.			

Se añade la entrada en la tabla contrata:

Data Output	Explain	Messages	Notifications
	<b>codigo_casting</b> [PK] character (10)	<b>codigo_candidato</b> [PK] character (10)	
1	ZDI2J077V0	TZCCRGP35Y	

### Tercer disparador:

```
-- TERCER TRIGGER
--Al insertar un candidato en la tabla "contrata" se debe comprobar si para ese casting en particular
--ya se han contratado suficientes personas. Para ello, se deberá comparar el número de candidatos
--seleccionados para ese casting con el número de personas requerido al contratar el casting.

create or replace function fun_comprobar_contrata() returns Trigger as
$comprobarContratados$
declare
    num_personas_requeridas smallint;
    num_personas_contratadas smallint;
begin
    num_personas_requeridas = (select numero_de_personas from presencial
                               where codigo_casting = new.codigo_casting);
    num_personas_contratadas = (select count(*) from contrata
                                where codigo_casting = new.codigo_casting);

    if(num_personas_requeridas = num_personas_contratadas) then
        raise notice 'Ya se han contratado a todos los candidatos necesarios';
        return null;
    else
        return new;
    end if;
end
$comprobarContratados$
Language plpgsql;

create trigger tr_comprobar_contrata before insert on contrata
for each row
execute procedure fun_comprobar_contrata();
```

En este caso, el evento que hace que este disparador se active es la inserción en la tabla contrata. Al tratarse de que queremos evaluar la validez de los datos, el disparador se activa antes de la inserción y no después.

Al dispararse con la inserción de tuplas en la tabla contrata, el segundo disparador puede que active este.

Tenemos que comprobar que el número de personas que necesita el casting es superior al número actual de candidatos contratados (no queremos que el casting contrate personal de más). Por ello, en la función que se invoca cuando el disparador se activa, si ya no se necesitan más personas (primer if) devolvemos null y avisamos con un raise notice de tal manera que no se produzca la inserción en la tabla contrata. En caso contrario el casting necesita más candidatos (el else) y por lo tanto permitimos que se haga la inserción devolviendo new (la nueva tupla). Nunca se va a dar el caso de que tengamos más personas de las que necesitamos (debido al primer if) por ello basta con utilizar un if y un else y no se contempla otra posibilidad.



Comprobación:

```




--Prueba trigger 3:
-- El casting 'ZDI2J077V0' solo necesita 1 persona tal y como podemos comprobar en:
select numero_de_personas from presencial where codigo_casting = 'ZDI2J077V0';
-- Personas que tiene contratadas el casting (ya está lleno):
select * from contrata where codigo_casting = 'ZDI2J077V0';
-- Por lo que no deberíamos contratar más candidatos incluso si superan las pruebas:
-- (para hacer la prueba haremos un insert directamente)
insert into contrata values('ZDI2J077V0', 'NK9IRY891Y');
-- Comprobamos que no se ha añadido al casting 'ZDI2J077V0' (puede estar en otros)
select * from contrata where codigo_candidato = 'NK9IRY891Y';

```

Numero de personas que necesita el casting:

Data Output	Explain	Messages	Notifications
	numero_de_personas smallint		
1		1	

Personas que ya tiene contratadas:

Data Output	Explain	Messages	Notifications
	codigo_casting [PK] character (10)	 codigo_candidato [PK] character (10)	
1	ZDI2J077V0	TZCCRGP35Y	

Si intentamos hacer el insert:

Data Output	Explain	Messages	Notifications
NOTICE: Ya se han contratado a todos los candidatos necesarios INSERT 0 0  Query returned successfully in 42 msec.			

Comprobamos que la tupla que hemos intentado insertar no está en la tabla contrata:  
('ZDI2J077V0', 'NK9IRY891Y')

	Data Output	Explain	Messages	Notifications
	<b>codigo_casting</b> [PK] character (10)		<b>codigo_candidato</b> [PK] character (10)	
1	9C2JA0T52E		NK9IRY891Y	

(Está en otro casting, pero no en el que hemos intentado meterlo estando ya lleno).

#### Cuarto disparador:

```
-- CUARTO TRIGGER
--Cuando un candidato realiza una prueba de un determinado casting,
--se debe comprobar si su perfil encaja en alguno de los perfiles requeridos por en dicho casting.

create function fun_comprobar_perfil() returns Trigger as
$comprobarPerfil$
declare
    perfil_buscado character(10);
    perfil_candidato character(10);
begin
    perfil_buscado = (select codigo_de_perfil from casting_necesita_perfil
                      where codigo_casting = new.codigo_casting);
    perfil_candidato = (select codigo_de_perfil from candidatos
                       where codigo_candidato = new.codigo_candidato);

    if (perfil_buscado = perfil_candidato) then
        return new;
    else
        raise notice 'El candidato no puede realizar la prueba porque no encaja en ningún perfil';
        return null;
    end if;
end
$comprobarPerfil$
Language plpgsql;

create trigger tr_comprobar_perfil before insert on candidato_realiza_prueba
for each row
execute procedure fun_comprobar_perfil();
```

Este disparador, al igual que el primero y el segundo, se activa mediante la inserción en la tabla candidato\_realiza\_prueba. En este caso pretendemos validar los datos por lo que, a diferencia de los dos primeros disparadores, este se activa antes de la inserción.

La función que invoca el disparador se basa en comparar el perfil buscado por el casting (localizado en la tabla casting\_necesita\_perfil) y el perfil que posee el candidato que pretende hacer una prueba para ese casting. Si ambos coinciden quiere decir que el casting está buscando a gente que sean como el candidato por lo que permitimos que realice la prueba (caso else, devolvemos new y permitimos la inserción). Si no es el caso, no le permitimos hacer la prueba porque incluso si las supera el casting no está interesado en contratar a candidatos de su perfil (caso del primer if, notificamos con un raise notice y no hacemos la inserción devolviendo null).



Comprobación:

```
--Prueba trigger 4:
-- El casting 'ZDI2J077V0' necesita del perfil:
select codigo_de_perfil from casting_necesita_perfil where codigo_casting = 'ZDI2J077V0';
-- Por lo que el candidato '4QU32DOAEZ' no puede hacer pruebas para ese casting ya
-- que tiene otro perfil, tal y como podemos ver en el siguiente select:
select codigo_de_perfil from candidatos where codigo_candidato = '4QU32DOAEZ';
-- Comprobacion trigger:
insert into candidato_realiza_prueba values('4QU32DOAEZ', '1', 'LMBJHFWLQE', 'ZDI2J077V0', 'TRUE');
-- Comprobamos que no se ha añadido
select * from candidato_realiza_prueba where
codigo_candidato = '4QU32DOAEZ' and numero = '1' and codigo_fase = 'LMBJHFWLQE'
and codigo_casting = 'ZDI2J077V0';
```

Perfil que busca el casting:

Data Output	Explain	Me
<div><div></div><div><div>codigo_de_perfil</div><div>character (10)</div></div><div></div></div>		
1	QAU8GWQRXD	

Perfil que tiene el candidato que pretendemos hacer que participe en el casting:

Data Output	Explain	Me
<div><div></div><div><div>codigo_de_perfil</div><div>character (10)</div></div><div></div></div>		
1	4FH8OJER99	

Dado que son diferentes no podremos hacer la inserción:

Data Output	Explain	Messages	Notifications
NOTICE: El candidato no puede realizar la prueba porque no encaja en ningún perfil INSERT 0 0			
Query returned successfully in 35 msec.			

Comprobación de que no se ha añadido:

Data Output	Explain	Messages	Notifications	
<div><div></div><div><div>codigo_candidato</div><div>[PK] character (10)</div></div><div></div></div>	<div><div></div><div><div>numero</div><div>[PK] smallint</div></div><div></div></div>	<div><div></div><div><div>codigo_fase</div><div>[PK] character (10)</div></div><div></div></div>	<div><div></div><div><div>codigo_casting</div><div>[PK] character (10)</div></div><div></div></div>	<div><div></div><div><div>resultado_prueba</div><div>boolean</div></div><div></div></div>

## Acceso desde Java

La aplicación JavaDatabase se desarrolla en 3 fases:

1º Autenticación del usuario con contraseña. Cuando se introduzcan el programa intentará conectarse a la base de datos PL3.

```
Inserte el usuario con el que se quiere conectar:
jefe
Inserte la contraseña:
jefe
=====
Conectandose a la base de datos PL3
=====
Conectado!!
```

2º Consultas fijas. El usuario podrá elegir entre 20 queries introduciendo en el sistema un número entre el 1 y el 20 que ejecutarán y mostrarán por pantalla los resultados tras buscarlos en la base de datos. Esta fase se encuentra en un bucle el cual se terminará de ejecutar tras introducir la opción 21 en el programa.

```
1*- Mostrar el nombre de los clientes que no han contratado ningún casting.
2*- Mostrar el nombre de los candidatos que han superado todas las pruebas
3*- Mostrar el número de candidatos que hay asociados a cada perfil.
4*- Mostrar el nombre del empleado que más castings ha dirigido.
5*- Mostrar el número de candidatos que se han presentado a cada casting
6*- Mostrar el nombre y la dirección de las candidatas que tengan el pelo rubio y sean de Madrid
7*- Mostrar el código de perfil de los perfiles requeridos en los castings que incluyen la subcadena "anuncio tv" en su descripción.
8*- Mostrar el código de los candidatos que tienen representante y tienen el pelo castaño.
9*- Mostrar el precio total que ha de pagar cada candidato
10*- Mostrar el número de candidatos adultos y el número de candidatos niños que hay en la base de datos
11*- Mostrar el dni del agente que ha dirigido el casting en el que alguna prueba individual se ha llevado a cabo en la sala "flor"
12*- Mostrar la plataforma web que se ha usado en el casting online más caro, así como el nombre del cliente que ha contratado dicho casting.
13*- Mostrar el porcentaje de clientes que hay de cada tipo
14*- Mostrar el nombre de los candidatos que han superado alguna prueba de algún casting, así como el nombre del casting.
15*- Mostrar el dinero total recaudado por la empresa
16*- Mostrar el nombre y el teléfono de los representantes que representen a 2 candidatos como mínimo.
17*- Mostrar el dni de los adultos que no tengan representante
18*- Mostrar los datos del perfil más demandado así como el nombre del cliente que lo ha requerido para su casting.
19*- Mostrar el número de pruebas superadas por cada niño.
20*- Mostrar el código del casting que más fases tiene.
21*- Salir del Programa
Introduzca del listado anterior la query que quiere realizar:
```

## Ejemplo de ejecución de algunas instrucciones SQL:

```
Introduzca del listado anterior la query que quiere realizar:
1
Listado de Clientes que no han contratado ningun Casting
Cliente: Jesus.
Cliente: Antonio.
Cliente: Luis.
Cliente: Andrea.
Cliente: Vicente.
Cliente: Helena.
Cliente: Álvaro.
=====
Introduzca del listado anterior la query que quiere realizar:
18
Listado de perfiles más demandados junto a sus respectivos clientes
Nombre Cliente: Javier,Codigo Perfil: 4FH8OJER99, Provincia: Girona, Sexo: M, Altura: 184, Edad: 25, Color de Pelo: Castaño, Color de ojos: Castaños, Especialidad: Actor, Experiencia: t,
Nombre Cliente: Jorge, Codigo Perfil: 4FH8OJER99, Provincia: Girona, Sexo: M, Altura: 184, Edad: 25, Color de Pelo: Castaño, Color de ojos: Castaños, Especialidad: Actor, Experiencia: t,
Nombre Cliente: Carlos, Codigo Perfil: 4FH8OJER99, Provincia: Girona, Sexo: M, Altura: 184, Edad: 25, Color de Pelo: Castaño, Color de ojos: Castaños, Especialidad: Actor, Experiencia: t,
=====
Introduzca del listado anterior la query que quiere realizar:
3
Número de candidatos asociados a cada perfil
Codigo de perfil: F1SHW011FX, Numero de Candidatos: 1.
Codigo de perfil: 4FH8OJER99, Numero de Candidatos: 1.
Codigo de perfil: ZFMIOI42PS, Numero de Candidatos: 1.
Codigo de perfil: QAU8GWQRXD, Numero de Candidatos: 1.
Codigo de perfil: J4R9OQEUV3, Numero de Candidatos: 1.
Codigo de perfil: 5R7IU58U1Z, Numero de Candidatos: 1.
Codigo de perfil: DD1IYABIES, Numero de Candidatos: 1.
Codigo de perfil: 29T06C7ALK, Numero de Candidatos: 1.
Codigo de perfil: KPW6TUZKDQ, Numero de Candidatos: 1.
Codigo de perfil: 56GH5Y203E, Numero de Candidatos: 1.
=====
```

3º Desconexión y fin del programa.

```
Introduzca del listado anterior la query que quiere realizar:  
21  
Desconectandonos de la base de datos.  
Desconectado!!  
BUILD SUCCESSFUL (total time: 1 minute 3 seconds)
```