



Universidad
de Alcalá

PECL3 BBDD

Alcalá de Henares, Madrid, 29 de Enero de 2021

**Universidad de Alcalá
Escuela Politécnica
Grado en Ingeniería de Computadores
2020/2021**

Soria Parra, Alejandro DNI: 03213414S

Varona Rabasco, Eduardo DNI: 53841619E

Munuera Pérez, César DNI: 09105135X

Horario de laboratorio: 12:00-14:00

Tutor: Lino González García. Departamento Ciencias de la
Computación. Escuela Politécnica UAH

Contenido

Fase 1	2
Diccionario de datos	2
Modelo Entidad-Relación Extendido	4
Fase 2	6
Transformación del modelo ER al modelo relacional	6
Modelo relacional en Pgmodeler	7
Consultas SQL	8
Documentación detallada de la Fase 2	16
Fase 3	18
Normalización	18
Creación de roles y usuarios	29
Implementación de disparadores	37
Acceso desde Java	40
Documentación detallada.	43

Fase 1

Diccionario de datos

Entidad	Atributo	Dominio	Restricción
Cliente	Código Nombre Dirección Teléfono Contacto Tipo de actividad	Int Char [40] ----- Long Char [40] Bool	0 <= 9999 PK * Compuesto Multivaluado (Móvil, Fijo) ** * Restringido (Publicidad y cine / Moda)
Casting	Código Casting Nombre Descripción Fecha Contrato NºPersonas	Int Char [40] Char [50] Date Int	0 <= 9999 PK * * dd/mm/aaaa
Cast. Online	Fecha Plataforma web	Date Char [100]	dd/mm/aaaa URL
Cast. Presencial			
Agente de Casting	N.º de Empleado DNI Nombre Dirección	Int Char [9] Char [40] -----	0 <= 9999 CK 8 num y 1 letra PK * Compuesto
Candidatos	Código Nombre Dirección Teléfono Fecha Nac.	Int Char [40] ----- Long Date	PK * Compuesto Multivaluado (Móvil y Fijo) ** dd/mm/aaaa
Perfil	Código de perfil Provincia Sexo Altura Edad Color Pelo Color Ojos Especialidad Experiencia	Int Char [40] Bool Float Int Multivaluado Multivaluado Bool Bool	PK * Restringido (Hombre y Mujer) 0 < 3.00 0 < 95 Castaño, Negro, Rubio, otros Marron, Azul, Verde, Otros Restringido (Modelo o Actor) Si / No
Cand. Adulto	DNI	Char [9]	8 num y 1 letra CK

Cand. Niño	Nombre Tutor	Char [40]	*
Representante	NIF Nombre Teléfono Dirección	Char [9] Char [40] Long -----	8 num y 1 letra PK * Multivaluado (Móvil y Fijo) ** Compuesto
Fases	N.º Fase Fecha Inicio ID Fase	int Date Int	1 < 20 dd/mm/aaaa 0 <= 9999 PK
Pruebas Presenciales	N.º ID Fecha Sala Descripción	Int Date Int Char [50]	0 <= 9999 PK dd/mm/aaaa 0 <= 20 *
Pruebas online	NºID_PBA_ONLINE Fecha Sala_online Descripción	Int Date Int Char [50]	0 <= 9999 PK dd/mm/aaaa 0 <= 20 *

Relación	Entidades	Cardinalidad	Atributos
Contrata	Casting - Cliente	Casting 1,n Cliente 1,1 (1:N)	Coste
Dirigido	Cast. Presencial – Agente de Casting	Cast. Presencial 1,n Agente de Casting 1,1 (1:N)	
Entrevista	Agente de Casting - Candidato	Agente de Casting 1,1 Candidato 1,n (1:N)	
Corresponde	Candidato - Perfil	Candidato 1,1 Perfil 1,1 (1:1)	
Tiene	Candidato – Representante	Candidato 1,n Representante 0,1 (1:N)	
Consta de	Cast. Presencial - Fases	Cast. Presencial 1,1 Fases 1,N (1:N)	
Realiza	Candidato – Pruebas Presenciales	Candidato 1,n Pruebas Presenciales 1,n (N:M)	Coste Prueba
Se divide en	Fases – Pruebas presenciales	Fases 1,1 Pruebas presenciales 1,n (1:N)	
Consta de (2)	Casting Online - Pruebas Online	Casting Online 1,1 Pruebas Online 1,n (1:N)	

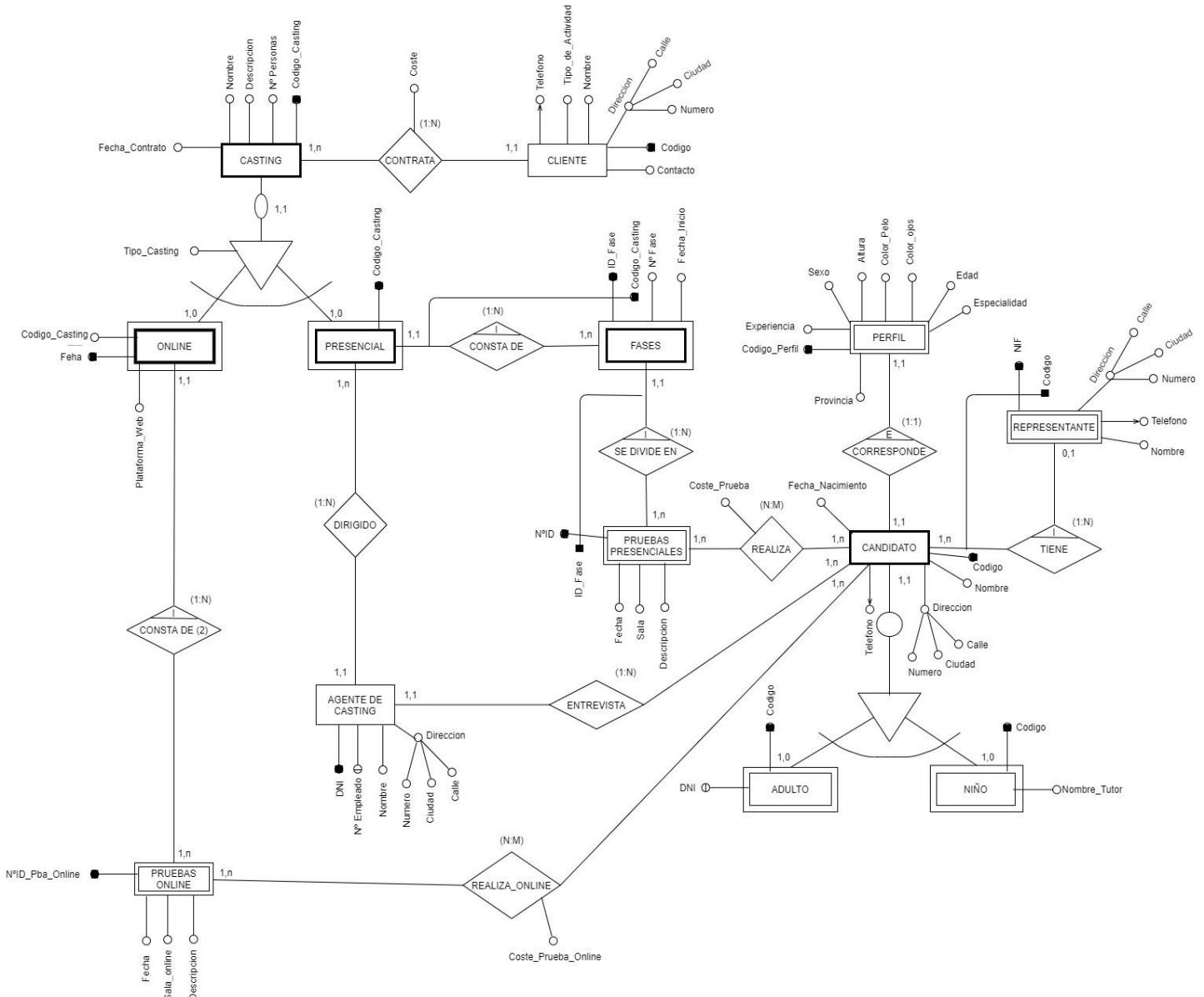
Realiza Online	Pruebas Online - Candidato	Pruebas Online 1,n Candidato 1,n (N:M)	Coste_Prueba_Online
----------------	----------------------------	--	---------------------

*Solo espacios y letras

** Solo son válidos números de 9 cifras que empiecen por 6, 7 o 9

Modelo Entidad-Relación Extendido

Diagrama hecho en Word, powerpoint, etc. No se admiten fotos de diagramas hechos a mano.



Documentación detallada de entidades y relaciones.

- Hemos puesto que el **Código del Candidato** sea de tipo long para mayor seguridad de los candidatos (de esta manera será más difícil que una tercera persona entre sin permiso al perfil de uno de nuestros candidatos).
- Hemos supuesto que los **Códigos/ID** nos los define la agencia y que tienen un formato específico, y este formato es de $0 \leq 9999$.
- Sobre la **Plataforma Web**, suponemos que la URL máxima es de 100 a pesar de que el tamaño máximo de una URL es de 2083 caracteres.
- Para los **Int** que no poseen restricción, es el propio **Int** el que se lo impone.
- El número máximo de **Salas** en las **Pruebas Presenciales** es de 20, ya que suponemos, que el límite nos lo define la agencia.
- El número máximo de **Salas Online** en las **Pruebas Online** es de 20, ya que suponemos, que el límite nos lo define la agencia (al igual que el límite de Salas para las Pruebas Presenciales).

Fase 2

Para esta fase hemos decidido usar el modelo resuelto que se nos ha proporcionado de la práctica 1.

Transformación del modelo ER al modelo relacional

Se indicarán **PKs** subrayadas, **FKs** en negrita y **PFKs** subrayadas y en negrita.

Agente de casting (numEmpleado, Nombre, dirección, DNI, **Código de casting**)

Casting Presencial (**Código de casting**)

Casting Online (**Código de casting**, plataforma web)

Casting (Nombre, Descripción, fecha, Coste, NumPersonas, Código de casting, **Código cliente**)

Casting_necesita_Perfil (**Código de casting**, **Código perfil**)

Perfil (**Código perfil**, Provincia, Sexo, Altura, Edad, Color ojos, Color pelo, Experiencia, Especialidad)

Cliente (Código cliente, Nombre, Calle, Portal, Piso, Ciudad, Tipo Actividad, Persona de contacto)

Telefono_cliente (Teléfono, **Código cliente**)

Cliente_contrata_Candidato (**Código cliente**, **Código candidato**, Casting)

Candidato (Código candidato, Nombre, Calle, Piso,Ciudad, Portal, Fecha nacimiento, **Código perfil**, **NIF de representante**,)

Candidato_Niño (**Código candidato**, nombre tutor)

Candidato_Adulto (**Código candidato**, DNI)

Teléfono candidato (Teléfono, **Código Candidato**)

Prueba individual_realiza_Candidato (**Código casting**, **Código fase**, **Código prueba**, **Código candidato**, válido)

Representante (**NIF**, Nombre)

Telefono Representante (Teléfono, **NIF Representante**)

Prueba individual (**Código Prueba**, **Código fase**, **Código fase**, Fecha, Sala, Descripción, Coste)

Fase (**Código fase**, **Código casting**, Fecha)

Consultas SQL

Se indicará el enunciado de cada consulta y a continuación se incluirá una captura de pantalla desde Pgadmin del código en sql junto a su correspondiente resultado.

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

```
1 --MOSTRAR LOS NOMBRES DE LOS CLIENTES QUE NO HAN CONTRATADO NINGUN CASTING
2
3 SELECT "Código_cliente", "Nombre"
4     FROM "Cliente"
5      WHERE "Cliente"."Código_cliente" NOT IN (SELECT "Casting_presencial"."Código_cliente_Cliente" FROM "Casting_presencial")
6 INTERSECT SELECT "Código_cliente", "Nombre"
7     FROM "Cliente"
8      WHERE "Cliente"."Código_cliente" NOT IN (SELECT "Casting_online"."Código_cliente_Cliente" FROM "Casting_online")
9
10
```

Data Output Explain Messages Notifications

Código_cliente	Nombre
4	Modas Alcarria
10	

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

```
0 -- 2. MOSTRAR EL NOMBRE DE LOS CANDIDATOS QUE HAYAN SUPERADO TODAS LAS PRUEBAS
1
2 SELECT "Candidato_adulto"."Nombre"
3     FROM "Candidato_adulto"
4      WHERE "Candidato_adulto"."Código_candidato" IN (SELECT "Código_candidato_Candidato_adulto"
5                                              FROM "Candidato_adulto_realiza_Prueba_individual" WHERE "Valido" = True)
6 UNION SELECT "Candidato_ninio"."Nombre"
7     FROM "Candidato_ninio"
8      WHERE "Candidato_ninio"."Código_candidato" IN (SELECT "Código_candidato_Candidato_ninio"
9                                              FROM "Candidato_ninio_realiza_Prueba_individual" WHERE "Valido" = True)
```

Data Output Explain Messages Notifications

Nombre
Florentino
Carlos
Carla
Juan
Kiko Rivera

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

```
19 -- 3. MOSTRAR EL NÚMERO DE CANDIDATOS QUE HAY ASOCIADOS A CADA PERFIL
20
21 SELECT COUNT("Codigo_perfil_Perfil") AS "Numero_Candidatos", "Codigo_perfil_Perfil"
22   FROM "Candidato_adulto"
23   GROUP BY "Codigo_perfil_Perfil"
24 UNION
25 SELECT COUNT("Codigo_perfil_Perfil") AS "Numero_Candidatos", "Codigo_perfil_Perfil"
26   FROM "Candidato_ninio"
27   GROUP BY "Codigo_perfil_Perfil"
28 ORDER BY "Codigo_perfil_Perfil"
29
30
31 -- 4. MOSTRAR EL NOMBRE DEL EMPLEADO QUE MÁS CASTINGS HA DIRIGIDO (DADO NUESTRO MODELO, ESTA CONSULTA NO SE PUEDE REALIZAR).
```

Data Output Explain Messages Notifications

	Numero_Candidatos bigint	Codigo_perfil_Perfil integer
1	1	0
2	2	1
3	1	2
4	1	3
5	2	4
6	1	5
7	1	6
8	4	10
9	1	11

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

Esta consulta no se ha podido realizar, debido a que el modelo empleado para esta fase de la práctica no contempla una consulta de este tipo, ya que la cardinalidad de la relación no lo permite. Sin embargo, aquí mostramos nuestro planteamiento de cómo podría realizarse dicha consulta, en el caso de que el modelo ER empleado tuviese una cardinalidad correcta.

```
SELECT "Nombre", COUNT("Código_casting_Casting_presencial") AS
"numero_codigos" FROM "Agente de casting" WHERE "numero_codigos"
= (SELECT MAX("numero_codigos") FROM "Agente de casting")
GROUP BY "Nombre" ORDER BY "Nombre"
```

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

```
33 -- 5. MOSTRAR EL NÚMERO DE CANDIDATOS QUE SE HAN PRESENTADO A CADA CASTING (QUE AL MENOS HAYAN REALIZADO UNA PRUEBA).
34
35 SELECT COUNT("Código_prueba_Prueba_individual") AS "Numero_Candidatos", "Código_candidato_Candidato_adulto" AS "Codigo_candidato"
36   FROM "Candidato_adulto_realiza_Prueba_individual"
37   GROUP BY "Código_candidato_Candidato_adulto"
38 UNION
39 SELECT COUNT("Código_prueba_Prueba_individual") AS "Numero_Candidatos", "Código_candidato_Candidato_ninio" AS "Codigo_candidato"
40   FROM "Candidato_ninio_realiza_Prueba_individual"
41   GROUP BY "Código_candidato_Candidato_ninio"
42 ORDER BY "Codigo_candidato"
```

Data Output Explain Messages Notifications

	Numero_Candidatos bigint	Codigo_candidato integer
1	1	0
2	1	2
3	1	3
4	1	4
5	1	2001
6	1	2002
7	2	2003
8	1	2005
9	1	2007

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

```

29 -- MOSTRAR EL NOMBRE Y LA DIRECCÓN DE LAS CANDIDATAS QUE TENGAN PELO RUBIO Y SEAN DE MADRID.
30
31 SELECT "Nombre", "Ciudad", "Calle", "Portal", "Piso"
32     FROM "Candidato_adulto" INNER JOIN "Perfil" ON "Candidato_adulto"."Codigo_perfil_Perfil" = "Perfil"."Codigo_perfil"
33         WHERE ("Ciudad" = 'Madrid') and ("Color_de_pelo" = 'Rubio') and ("Sexo" = False)
34 UNION SELECT "Nombre", "Ciudad", "Calle", "Portal", "Piso"
35     FROM "Candidato_ninio" INNER JOIN "Perfil" ON "Candidato_ninio"."Codigo_perfil_Perfil" = "Perfil"."Codigo_perfil"
36         WHERE ("Ciudad" = 'Madrid') and ("Color_de_pelo" = 'Rubio') and ("Sexo" = False)
37

```

Data Output Explain Messages Notifications					
	Nombre character varying (40)	Ciudad character varying (20)	Calle character varying (20)	Portal character varying (20)	Piso character varying (20)
1	Carla	Madrid	Calle Melancolia	24	3ºA
2	Carmen	Madrid	Calle La Calle	2	6ºA

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

```

38 -- 7. MOSTRAR EL CÓDIGO DE PERFIL DE LOS PERFILES REQUERIDOS EN LOS CASTING QUE INCLUYEN LA SUBCADENA 'ANUNCIO TV' EN SU DESCRIPCIÓN.
39
40 SELECT "Codigo_perfil_Perfil", "Descripción"
41     FROM "Casting_presencial_necesita_Perfil" INNER JOIN "Casting_presencial"
42         ON "Casting_presencial_necesita_Perfil"."Código_casting_Casting_presencial" = "Casting_presencial"."Código_casting"
43             WHERE "Casting_presencial"."Descripción" LIKE '%Anuncio TV%'
44 UNION SELECT "Codigo_perfil_Perfil", "Descripción"
45     FROM "Casting_online_necesita_Perfil" INNER JOIN "Casting_online"
46         ON "Casting_online_necesita_Perfil"."Código_casting_Casting_online" = "Casting_online"."Código_casting"
47             WHERE "Casting_online"."Descripción" LIKE '%Anuncio TV%'
48
49 -- 8. MOSTRAR EL CÓDIGO DE LOS CANDIDATOS QUE TIENEN REPRESENTANTE Y TIENEN PELO 'Castanio'
50
51 SELECT "Codigo_perfil_Perfil"

```

Data Output Explain Messages Notifications		
	Codigo_perfil_Perfil integer	Descripción character varying (50)
1		8 Anuncio TV

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

```

51 SELECT "Código candidato"
52     FROM "Candidato_adulto" INNER JOIN "Perfil"
53         ON "Candidato_adulto"."Codigo_perfil_Perfil" = "Perfil"."Codigo_perfil"
54             WHERE ("NIF_Representante" IS NULL) AND ("Color_de_pelo" = 'Castanio')
55 UNION SELECT "Código candidato"
56     FROM "Candidato_ninio" INNER JOIN "Perfil"
57         ON "Candidato_ninio"."Codigo_perfil_Perfil" = "Perfil"."Codigo_perfil"
58             WHERE ("NIF_Representante" IS NULL) AND ("Color_de_pelo" = 'Castanio')
59
60
61
62

```

Data Output Explain Messages Notifications			
	Código candidato [PK] integer		
1	2008		

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

```

59 -- 9. MOSTRAR EL PRECIO TOTAL QUE HA DE PAGAR CADA CANDIDATO.
60 -- (HAY QUE DARLE UNA PENSADA PORQUE LOS CANDIDATOS QUE HAYAN REALIZADO VARIOS CASTINGS Y VARIAS PRUEBAS ONLINE TENDRÁN VARIA TUPLAS CON EL COSTE PARA CADA UNA)
61 /*CREAMOS UNA VISTA QUE AGRUPE LOS CODIGOS DE LOS CANDIDATOS CON LO QUE TIENEN QUE PAGAR POR CADA UNA DE LAS PRUEBAS QUE HAN REALIZADO*/
62 CREATE VIEW "Pagos_candidatos" AS
63 SELECT "Código candidato_Candidato_adulto", T1."Coste"
64   FROM ("Candidato_adulto_realiza_Prueba_individual" INNER JOIN "Prueba_individual"
65     ON "Candidato_adulto_realiza_Prueba_individual"."Código_prueba_Prueba_individual" = "Prueba_individual"."Código_prueba") as T1
66 UNION SELECT "Código candidato_Candidato_ninio", T1."Coste" as Coste_total
67   FROM ("Candidato_ninio_realiza_Prueba_individual" INNER JOIN "Prueba_individual"
68     ON "Candidato_ninio_realiza_Prueba_individual"."Código_prueba_Prueba_individual" = "Prueba_individual"."Código_prueba") as T1
69       ORDER BY "Código candidato_Candidato_adulto"
70 /*CONSULTA*/
71 SELECT T1."Código candidato_Candidato_adulto" AS "Codigo_candidatos", T1."Coste_total"
72   FROM (SELECT "Código candidato_Candidato_adulto", SUM("Coste") AS "Coste_total"
73     FROM "Pagos_candidatos" WHERE "Código candidato_Candidato_adulto" = "Código candidato_Candidato_adulto"
74       GROUP BY "Código candidato_Candidato_adulto") AS T1
75
76

```

Data Output Explain Messages Notifications		
	Codigo_candidatos integer	Coste_total double precision
1	0	7.8
2	2	6.5
3	3	5.5
4	4	17.9
5	2001	10
6	2002	11.6
7	2003	12
8	2005	1.5
9	2007	5.5

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

```

73 -- 10. MOSTRAR EL NÚMERO DE CANDIDATOS ADULTOS Y EL NÚMERO DE CANDIDATOS NIÑOS QUE HAY EN LA BASE DE DATOS.
74
75 SELECT COUNT ("Candidato_adulto"."Código candidato")
76   AS Numero_candidatos_adultos, (SELECT COUNT ("Candidato_ninio"."Código candidato") as Numero_candidatos_ninos
77     FROM "Candidato_ninio")
78   FROM "Candidato_adulto"
79
80
81
82

```

Data Output Explain Messages Notifications		
	numero_candidatos_adultos bigint	numero_candidatos_ninos bigint
1	9	5

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”.

```

80 -- 11. MOSTRAR EL DNI DE LA GENTE QUE HA DIRIGIDO EL CASTING EN EL QUE ALGUNA PRUEBA INDIVIDUAL SE HA LLEVADO A CABO EN LA SALA 'flor'
81
82 SELECT "DNI"
83   FROM ("Agente_de_casting" INNER JOIN "Prueba_individual"
84     ON "Agente_de_casting"."Código_casting_Casting_presencial" = "Prueba_individual"."Código_casting_Casting_presencial_Fase")
85   WHERE "Prueba_individual"."Sala" = 'flor'
86
87
88
89
Data Output Explain Messages Notifications


| DNI | character (9) |
|-----|---------------|
| 1   | 02030488A     |


```

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.

```

88 -- 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
89
90 SELECT "Plataforma_web", "Cliente"."Nombre"
91   FROM ("Casting_online" INNER JOIN "Cliente"
92     ON "Casting_online"."Código_cliente_Cliente" = "Cliente"."Código_cliente")
93   WHERE "Casting_online"."Coste" = (SELECT MAX("Coste") FROM "Casting_online")
94
95
96
97
98
Data Output Explain Messages Notifications


| Plataforma_web            | Nombre          |
|---------------------------|-----------------|
| 1 http://messenger.com/27 | El Coche Ingles |


```

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

```

10
11 Query Editor Query History
1 -- 13. Mostrar el porcentaje de clientes que hay de cada tipo
2 CREATE VIEW "Total_clientes"
3   AS SELECT (SELECT COUNT ("Tipo_actividad") FROM "Cliente") as "Clientes_totales",
4   (SELECT COUNT ("Tipo_actividad") FROM "Cliente" WHERE "Cliente"."Tipo_actividad" = True) as "Clientes_Empresa_de_moda",
5   (SELECT COUNT ("Tipo_actividad") FROM "Cliente" WHERE "Cliente"."Tipo_actividad" = False) as "Clientes_Empresa_de_publicidad_y_cine"
6
7 SELECT ((SELECT CAST("Clientes_Empresa_de_moda" AS FLOAT) FROM "Total_clientes") / (SELECT "Clientes_totales" FROM "Total_clientes"))*100
8 AS "Porcentaje_empresas_moda", (((SELECT CAST("Clientes_Empresa_de_publicidad_y_cine" AS FLOAT)
9                               FROM "Total_clientes") / (SELECT "Clientes_totales" FROM "Total_clientes"))*100) AS "Porcentaje_empresas_publicidad"
10
Data Output Explain Messages Notifications


| Porcentaje_empresas_moda | Porcentaje_empresas_publicidad |
|--------------------------|--------------------------------|
| 1 72.72727272727273      | 27.27272727272727              |


```

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

```

100
101 -- 14. MOSTRAR EL NOMBRE DE LOS CANDIDATOS QUE HAN SUPERADO ALGUNA PRUEBA DE ALGÚN CASTING, ASÍ COMO EL NOMBRE DEL CASTING
102 /*CREAMOS UNA VISTA QUE RECOJA LOS DATOS DE LOS CANDIDATOS ADULTOS QUE NECESITAMOS*/
103 CREATE VIEW "Candidato_adulto_realiza3"(Nombre_candidato_adulto, "Código_casting", "Es_valido")
104     AS SELECT "Nombre", "Código_casting_Casting_presencial_Fase_Prueba_individual", "Valido"
105     FROM ("Candidato_adulto" INNER JOIN "Candidato_adulto_realiza_Prueba_individual"
106         ON "Candidato_adulto"."Código candidato" = "Candidato_adulto_realiza_Prueba_individual"."Código candidato_Candidato_adulto")
107 /*CREAMOS UNA VISTA QUE RECOJA LOS DATOS DE LOS CANDIDATOS NIÑOS QUE NECESITAMOS*/
108 CREATE VIEW "Candidato_ninio_realiza"(Nombre_candidato_ninio, "Código_casting", "Es_valido")
109     AS SELECT "Nombre", "Código_casting_Casting_presencial_Fase_Prueba_individual", "Valido"
110     FROM ("Candidato_ninio" INNER JOIN "Candidato_ninio_realiza_Prueba_individual"
111         ON "Candidato_ninio"."Código candidato" = "Candidato_ninio_realiza_Prueba_individual"."Código candidato_Candidato_ninio")
112 /*CONSULTA*/
113 SELECT nombre_candidato_adulto AS Nombre_candidatos, "Nombre" AS Nombre_casting
114     FROM ("Candidato_adulto_realiza3" NATURAL INNER JOIN "Casting_presencial")
115     WHERE "Es_valido" = True
116 UNION
117 SELECT nombre_candidato_ninio, "Nombre"
118     FROM ("Candidato_ninio_realiza" NATURAL INNER JOIN "Casting_presencial")
119     WHERE "Es_valido" = True
120
121
122
123
124
125
126
```

Data Output Explain Messages Notifications

nombre_candidatos	nombre_casting
Carlos	C.Cine
Florentino	C.Construccores
Kiko Rivera	C.Antena10
Juan	C.Rosas
Carla	C.Antena7

15. Mostrar el dinero total recaudado por la empresa.

```

16 -- 15. MOSTRAR EL DINERO TOTAL RECAUDADO POR LA EMPRESA
17
18 /*REUTILIZAMOS LA VISTA CREADA EN LA CONSULTA 9. CON EL PRECIO QUE TIENE QUE PAGAR CADA
19 CANDIDATO (PARA SABER LO QUE LA EMPRESA RECAUDA POR LOS CANDIDATOS)*/
20 CREATE VIEW "Pagos_candidatos" AS
21 SELECT "Código candidato_Candidato_adulto", T1."Coste"
22     FROM ("Candidato_adulto_realiza_Prueba_individual" INNER JOIN "Prueba_individual"
23         ON "Candidato_adulto_realiza_Prueba_individual"."Código_prueba_Prueba_individual" = "Prueba_individual"."Código_prueba") as T1
24 UNION SELECT "Código candidato_Candidato_ninio", T1."Coste" as Coste_total
25     FROM ("Candidato_ninio_realiza_Prueba_individual" INNER JOIN "Prueba_individual"
26         ON "Candidato_ninio_realiza_Prueba_individual"."Código_prueba_Prueba_individual" = "Prueba_individual"."Código_prueba") as T1
27     ORDER BY "Código candidato_Candidato_adulto"
28
29
30 /*CREAMOS UNA TABLA CON LOS COSTES QUE TIENEN QUE PAGAR LOS CLIENTES POR LOS CASTINGS ONLINE Y PRESENCIALES*/
31 CREATE VIEW "Pagos_clientes" AS
32 SELECT "Código_cliente_Cliente", "Coste"
33     FROM "Casting_online"
34 UNION SELECT "Código_cliente_Cliente", "Coste"
35     FROM "Casting_presencial"
36
37 /*CREAMOS OTRA VISTA CON LOS COSTES EN UNA ÚNICA COLUMNA PARA PODER UTILIZAR
38 FUNCIONES AGREGADAS SOBRE EL RESULTADO DE OTRAS FUNCIONES AGREGADAS*/
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
```

Data Output Explain Messages Notifications

```

155 /*CREAMOS OTRA VISTA CON LOS COSTES EN UNA ÚNICA COLUMNA PARA PODER UTILIZAR
156 FUNCIONES AGREGADAS SOBRE EL RESULTADO DE OTRAS FUNCIONES AGREGADAS*/
157 CREATE VIEW "COSTE_TOTAL_CLIENTES_CANDIDATOS" AS
158 SELECT SUM("Coste") FROM "Pagos_clientes" UNION SELECT SUM("Coste") FROM "Pagos_candidatos"
159
160
161 /*CONSULTA*/
162 SELECT SUM("sum") FROM "COSTE_TOTAL_CLIENTES_CANDIDATOS"
163
164
```

Data Output Explain Messages Notifications

sum
double precision
393.40000000000003

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

```
0 -- 16. MOSTRAR EL NOMBRE Y EL TELEFONO DE LOS REPRESENTANTES QUE REPRESENTEN A DOS CANDIDATOS COMO MINIMO
0
1 /*CREAMOS UNA VISTA PARA GUARDAR EL INNER JOIN ENTRE Representante Y SU Telefono (ATRIBUTO MULTIVALUADO)*/
2 CREATE VIEW "Representante_con_telefono1" ("NIF", "Nombre_representante", "Telefono")AS
3 SELECT "NIF", "Nombre", "Telefono"
4 FROM "Representante" INNER JOIN "Telefono_representante" ON "Representante"."NIF" = "Telefono_representante"."NIF_Representante"
5 -- "NIF_Representante es una errata arrastrada desde PGModeler"
6
7 /*CONSULTA*/
8 SELECT DISTINCT "Nombre_representante", "Telefono", COUNT("Código candidato")
9   FROM "Representante_con_telefono1" INNER JOIN "Candidato_adulto" ON
10    "Candidato_adulto"."NIF_Representante" = "Representante_con_telefono1"."NIF"
11   GROUP BY "Representante_con_telefono1"."Nombre_representante", "Representante_con_telefono1"."Telefono"
12   HAVING COUNT("Código candidato") >= 2
13 UNION
14 SELECT DISTINCT "Nombre_representante", "Telefono", COUNT("Código candidato")
15   FROM "Representante_con_telefono1" INNER JOIN "Candidato_ninio" ON
16    "Candidato_ninio"."NIF_Representante" = "Representante_con_telefono1"."NIF"
17   GROUP BY "Representante_con_telefono1"."Nombre_representante", "Representante_con_telefono1"."Telefono"
18   HAVING COUNT("Código candidato") >= 2
19 ORDER BY "Nombre_representante"
0
```

sta Output Explain Messages Notifications

Nombre_representante	Telefono	count
Chewbacca	630887137	2
Organa	698707587	2
Organa	621842233	2

Organa sale 2 veces ya que tiene 2 teléfonos.

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

```
195 -- 17. MOSTRAR EL DNI DE LOS ADULTOS QUE NO TENGAN REPRESENTANTE.
196
197 SELECT "DNI"
198 FROM "Candidato_adulto"
199 WHERE "Candidato_adulto"."NIF_Representante" IS NULL
200
```

Data Output Explain Messages Notifications

DNI
character (9)
1 30235565B

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

```
-- 18. MOSTRAR LOS DATOS DEL PERFIL MÁS DEMANDADO ASÍ COMO EL NOMBRE DEL CLIENTE QUE LO HA REQUERIDO PARA SU CASTING (INCOMPLETO).

CREATE VIEW "Perfil_mas_demandado_casting_online" ("Nombre_cliente", "Código_casting", "Codigo_perfil_Perfil") AS
SELECT "Nombre", "Código_casting", "Codigo_perfil_Perfil"
FROM "Cliente" INNER JOIN "Casting_online" ON "Cliente"."Código_cliente" = "Casting_online"."Código_cliente_Cliente" AS T1
INNER JOIN "Casting_online_necesita_Perfil" ON T1."Código_casting" = "Casting_online_necesita_Perfil"."Código_casting_Casting_online"

/*CREAMOS VISTA PARA AGRUPAR EL NÚMERO DE FASES QUE TIENE CADA CASTING
CON SU CÓDIGO DE CASTING PRESENCIAL Y ASÍ PODER USAR LAS FUNCIONES AGREGADAS*/
CREATE VIEW "Cliente_casting_online_necesita_perfil" AS
SELECT COUNT("Código_de_fase") "Número_de_fases", "Código_casting_Casting_presencial"
FROM "Fase"
GROUP BY "Código_casting_Casting_presencial"

/*CONSULTA*/
SELECT "Código_casting_Casting_presencial", "Número_de_fases"
FROM "Num_fases_por_casting_presencial3"
WHERE "Número_de_fases" = (SELECT MAX("Número_de_fases") FROM "Num_fases_por_casting_presencial3")
```

Código_casting_Casting_presencial	Número_de_fases
1	2

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

```
218 -- 19. MOSTRAR EL NÚMERO DE PRUEBAS SUPERADAS POR CADA NIÑO.
219
220 SELECT COUNT("Valido"), "Código_candidato_Candidato_ninio"
221 FROM "Candidato_ninio_realiza_Prueba_individual"
222 WHERE "Candidato_ninio_realiza_Prueba_individual"."Valido" = True
223 GROUP BY "Código_candidato_Candidato_ninio"
224
225 -- 20. MOSTRAR EL CÓDIGO DEL CASTING QUE MÁS FASES TIENE.
226
```

count	Código_candidato_Candidato_ninio
1	4
2	0

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

```
-- 20. MOSTRAR EL CÓDIGO DEL CASTING QUE MÁS FASES TIENE.

/*CREAMOS VISTA PARA AGRUPAR EL NÚMERO DE FASES QUE TIENE CADA CASTING
CON SU CÓDIGO DE CASTING PRESENCIAL Y ASÍ PODER USAR LAS FUNCIONES AGREGADAS*/
CREATE VIEW "Num_fases_por_casting_presencial3" AS
SELECT COUNT("Código_de_fase") "Número_de_fases", "Código_casting_Casting_presencial"
FROM "Fase"
GROUP BY "Código_casting_Casting_presencial"

/*CONSULTA*/
SELECT "Código_casting_Casting_presencial", "Número_de_fases"
FROM "Num_fases_por_casting_presencial3"
WHERE "Número_de_fases" = (SELECT MAX("Número_de_fases") FROM "Num_fases_por_casting_presencial3")
```

Código_casting_Casting_presencial	Número_de_fases
1	2

Documentación detallada de la Fase 2

- **Problema 1:** Una vez finalizado el modelo relacional, exportarlo a PGAdmin4 e instanciar las relaciones, nos dimos cuenta de que una de las tablas (la tabla Telefono de Candidato Adulto, que se corresponde con el atributo multivaluado Telefono de Candidato Adulto) no estaba implementada, por lo que nos vimos en la necesidad de crear una nueva tabla dentro de PGAdmin4 para solucionar dicho error.
- **Problema 2:** La consulta número cuatro no se ha podido implementar debido a que la cardinalidad de las entidades participantes, en dicha consulta, es incongruente con el modelo ER del que decidimos partir a la segunda fase de esta práctica. No en vano, hemos propuesto una posible solución a dicha consulta, a pesar de no poder realizar su comprobación.
- **Cuestiones con el atributo Color_de_pelo:** Este atributo finalmente no se considerará como multivaluado (de cara al modelo relacional), pues para que así fuera, la relación entre Perfil y Color_de_pelo debería ser "uno-muchos", y a nuestro modo de ver, carecería de sentido buscar un único perfil que tuviera varios colores de pelo a la vez.

Pensando ya en el futuro programa que se nos pedirá hacer, hemos pensado que la manera de pedir este atributo será la siguiente: se pondrán una serie de restricciones al usuario para que no pueda introducir cualquier color de pelo, solo aquellos que estén contemplados en esta lista (castaño, rubio, negro, otros)

- **Cuestiones con el atributo Color_de_ojos:** Este atributo, al igual que el mencionado anteriormente, tampoco será multivaluado, pues para que así fuera, la relación entre Perfil y Color_de_ojos debería ser "uno-muchos", y a nuestro modo de ver, carecería de sentido buscar un perfil que tuviera varios colores de ojos a la vez.

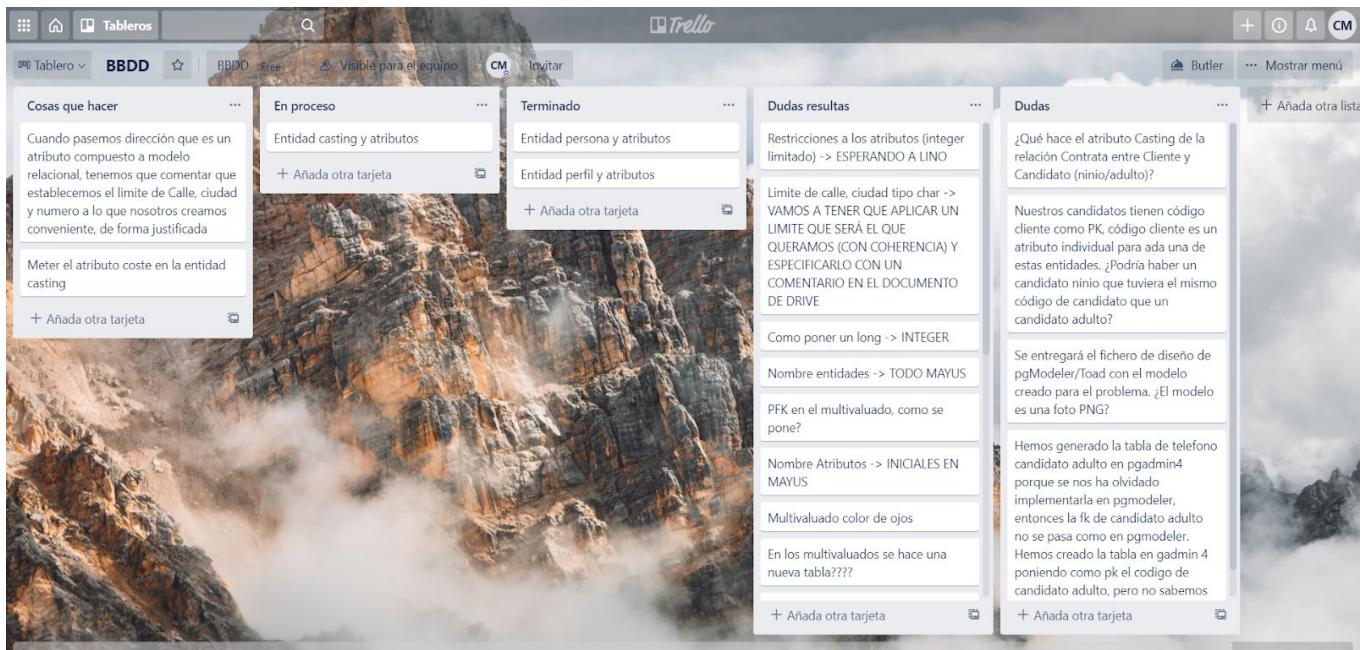
La manera de pedir este atributo una vez se haga el programa será la siguiente: se pondrán una serie de restricciones al usuario para que no pueda introducir cualquier color de ojos, solo aquellos que estén contemplados en esta lista (Marron, Azul, Verde, Otros)

- **Otras cuestiones menores:**

- Telefono_cliente: Empieza por 6, 7 o 9 pues podría tratarse también de un número fijo
- En el atributo portal, para aquellos que no residan en bloques de viviendas comunitarias este campo será null.

- Por otro lado, comentar la metodología de trabajo utilizada. Para realizar la práctica, hemos utilizado la plataforma *Discord* (videoconferencias), teniendo en cuenta los tiempos que corren.

Además, hemos utilizado la herramienta *Trello* para organizarnos y poner las cuestiones que nos iban surgiendo. Aquí una foto que muestra cómo la hemos empleado;



Fase 3

Normalización

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

En esta fase tenemos que comentar un problema surgido durante el desarrollo de la práctica. Lo primero que hicimos fue corregir los errores de la práctica anterior, y esto conlleva arreglar las tablas del pgModeler.

Después de corregir los problemas y normalizar, nos encontramos con que pgAdmin no nos implementaba la herencia. Entonces tuvimos que retroceder al modelo de la práctica anterior.

Los siguientes puntos muestran la normalización de la PL2 corregida. Después, mostraremos las diferencias entre ambos modelos, y la normalización de aquellas tablas no normalizadas en el modelo corregido.

- **Cliente** (Codigo_cliente, Nombre, Ciudad, Calle, Portal, Piso, Persona_de_contacto, Tipo_actividad)

(A, B, C, D, E, F, G, H, I)

Claves: A

A \rightarrow B, C, D, E, F, G, H, I

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Telefono_cliente** (Telefono, Codigo_cliente_Cliente)

(A, B)

Claves: A, B

No hay dependencias funcionales.

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos no primos tienen que depender totalmente de las claves. Entonces se cumple porque todos nuestros atributos son primos.

Una tabla está en 3^a FN, si los atributos no primos no dependen de forma transitiva de las claves. Entonces, en nuestro caso, está en 3^aFN ya que no tenemos atributos no primos.

- **Casting** (Codigo_casting, Nombre, Descripcion, Coste, NumPersonas, Fecha)
(A, B, C, D, E, F)

Claves: A

$A \rightarrow B, C, D, E, F$

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Casting_online** (Codigo_casting, Nombre, Descripcion, Coste, NumPersonas, Fecha, Plataforma_web)
(A, B, C, D, E, F, G)

Claves: A

$A \rightarrow B, C, D, E, F, G$

El único atributo añadido respecto a su padre es Plataforma_web, por ello depende del mismo PK.

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Casting_presencial** (Codigo_casting, Nombre, Descripcion, Coste, NumPersonas, Fecha)

(A, B, C, D, E, F)

Claves: A

$A \rightarrow B, C, D, E, F$

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Agente_de_casting** (numEmpleado, Nombre, DNI, Calle, Ciudad, Piso, Portal, Codigo_casting_Casting_presencial)
(A, B, C, D, E, F, G, H)

Claves: A

A \rightarrow B, C, D, E, F, G, H

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Fase** (Codigo_de_fase, Fecha, Codigo_casting_Casting_presencial)

(A, B, C)

Claves: A, C

A, C \rightarrow B

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Prueba_individual** (Codigo_prueba, Fecha, Sala, Descripcion, Coste, Codigo_de_fase_Fase, Codigo_casting_Casting_presencial_Fase)

(A, B, C, D, E, F, G)

Claves: A, F, G

A, F, G \rightarrow B, C, D, E

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Candidato_realiza_Prueba_individual** (Codigo_prueba_Prueba_individual, Valido, Codigo_de_fase_Fase_Prueba_individual, Codigo_casting_Casting_presencial_Fase_Prueba_individual, Codigo_candidato_Candidato)

(A, B, C, D, E)

Claves: A, C, D, E

A, C, D, E → B

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Candidato** (Codigo_candidato, Nombre, Fecha_nacimiento ,Calle, Piso, Ciudad, Portal, Codigo_perfil_Perfil, NIF_Representante)

(A, B, C, D, E, F, G, H, I)

Claves: A, H, I

A → B, C, D, E, F, G, H, I

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Telefono_Candidato** (Telefono, Codigo_candidato_Candidato)

(A, B)

Claves: A, B

No hay dependencias funcionales.

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos no primos tienen que depender totalmente de las claves. Entonces se cumple porque todos nuestros atributos son primos.

Una tabla está en 3^a FN, si los atributos no primos no dependen de forma transitiva de las claves. Entonces, en nuestro caso, está en 3^aFN ya que no tenemos atributos no primos.

- **Perfil** (Codigo_perfil, Provincia, Sexo, Altura, Edad, Especialidad, Esperiencia, Color_de_pelo, Color_de_ojos)

(A, B, C, D, E, F, G, H, I)

Claves: A

A \rightarrow B, C, D, E, F, G, H, I

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Casting_necesita_Perfil** (Codigo_casting_Casting, Codigo_perfil_Perfil)

(A, B)

Claves: A, B

No hay dependencias funcionales.

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Candidato_adulto** (DNI, Codigo_candidato, Nombre, Fecha_nacimiento, Calle, Piso, Ciudad, Portal, Codigo_perfil_Perfil, NIF_Representante)

(A, B, C, D, E, F, G, H, I, J)

Claves: B, I, J

A \rightarrow B, C, D, E, F, G, H, I, J

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Candidato_ninio** (Nombre_tutor, Codigo_candidato, Nombre, Fecha_nacimiento, Calle, Piso, Ciudad, Portal, Codigo_perfil_Perfil, NIF_Representante)

(A, B, C, D, E, F, G, H, I, J)

Claves: B, I, J

A → B, C, D, E, F, G, H, I, J

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Representante** (NIF, Nombre)

(A, B)

Claves: A

A → B

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Telefono_representante** (Telefono, NIF_Representante)

(A, B)

Claves: A, B

No hay dependencias funcionales.

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos no primos tienen que depender totalmente de las claves. Entonces se cumple porque todos nuestros atributos son primos.

Una tabla está en 3^a FN, si los atributos no primos no dependen de forma transitiva de las claves. Entonces, en nuestro caso, está en 3^aFN ya que no tenemos atributos no primos.

- **Cliente_contrata_Candidato** (Codigo_cliente_Cliente, Codigo_candidato_Candidato, Casting)

(A, B, C)

Claves: A, B

A, B -> C

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

A continuación, mostraremos las diferencias entre ambos modelos;

- **Cliente_contrata_Candidato** (Modelo corregido)
 - **Cliente_Contrata_Candidato_adulto** (Código_cliente_Cliente, Código_candidato_Candidato_adulto, Casting)

(A, B, C)

Claves: A, B

A, B -> C

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- **Cliente_Contrata_Candidato_ninio**(Código_cliente_Cliente, Código_candidato_Candidato_ninio, Casting) Código

(A, B, C)

Claves: A, B

A, B \rightarrow C

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- Casting_necesita_Perfil (Modelo corregido)

- Casting_online_necesita_perfil (Código_casting_Casting_online, Código_perfil_Perfil)

(A, B)

Claves: A, B

No hay dependencias funcionales.

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos no primos tienen que depender totalmente de las claves. Entonces se cumple porque todos nuestros atributos son primos.

Una tabla está en 3^a FN, si los atributos no primos no dependen de forma transitiva de las claves. Entonces, en nuestro caso, está en 3^aFN ya que no tenemos atributos no primos.

- Casting_presencial_necesita_perfil (Código_casting_Casting_presencial, Código_perfil_Perfil)

(A, B)

Claves: A, B

No hay dependencias funcionales.

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos no primos tienen que depender totalmente de las claves. Entonces se cumple porque todos nuestros atributos son primos.

Una tabla está en 3^a FN, si los atributos no primos no dependen de forma transitiva de las claves. Entonces, en nuestro caso, está en 3^aFN ya que no tenemos atributos no primos.

- Telefono_Candidato (Modelo corregido)

- Telefono_Candidato_ninio (Telefono, Código_candidato_Candidato_ninio)

(A, B)

Claves: A, B

No hay dependencias funcionales.

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos no primos tienen que depender totalmente de las claves. Entonces se cumple porque todos nuestros atributos son primos.

Una tabla está en 3^a FN, si los atributos no primos no dependen de forma transitiva de las claves. Entonces, en nuestro caso, está en 3^aFN ya que no tenemos atributos no primos.

- Telefono_Candidato_adulto (Telefono, Código_candidato_Candidato_adulto)

(A, B)

Claves: A, B

No hay dependencias funcionales.

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos no primos tienen que depender totalmente de las claves. Entonces se cumple porque todos nuestros atributos son primos.

Una tabla está en 3^a FN, si los atributos no primos no dependen de forma transitiva de las claves. Entonces, en nuestro caso, está en 3^aFN ya que no tenemos atributos no primos.

- Candidato (Modelo corregido)

- Candidato_adulto (Código_candidato, Nombre, Fecha_nacimiento, DNI, NIF_Representante, Código_perfil_Perfil, Calle, Piso, Ciudad, Portal)

(A, B, C, D, E, F, G, H, I, J)

Claves: A

A -> B, C, D, E, F, G, H, I, J

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- Candidato_ninio (Código_candidato, Nombre, Fecha_nacimiento, Nombre_tutor, NIF_Representante, Código_perfil_Perfil, Calle, Piso, Ciudad, Portal)

(A, B, C, D, E, F, G, H, I, J)

Claves: A

A -> B, C, D, E, F, G, H, I, J

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- Candidato_realiza_Prueba_individual (Modelo corregido)

- Candidato_adulto_realiza_Prueba_individual
(Código_prueba_Prueba_individual, Código_de_fase_Fase_Prueba_individual, Código_casting_Casting_presencial_Fase_Prueba_individual, Código_candidato_Candidato_adulto, Valido)

(A, B, C, D, E)

Claves: A, B, C, D

A, B, C, D -> E

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- Candidato_ninio_realiza_Prueba_individual
 (Código_prueba_Prueba_individual, Código_de_fase_Fase_Prueba_individual,
 Código_casting_Casting_presencial_Fase_Prueba_individual, Código_candidato_Candidato_ninio, Valido)

(A, B, C, D, E)

Claves: A, B, C, D

A, B, C, D -> E

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- Casting (Modelo corregido)

- Casting_presencial (Código_casting, Nombre, Descripción, Coste, NumPersonas, Fecha, Código_cliente_Cliente)

(A, B, C, D, E, F, G)

Claves: A

A -> B, C, D, E, F, G

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

- Casting_online (Código_casting, Nombre, Descripción, Coste, NumPersonas, Fecha, Código_cliente_Cliente, Plataforma_Web)

(A, B, C, D, E, F, G, H)

Claves: A

A -> B, C, D, E, F, G, H

Está en 1FN, ya que todos los dominios de los atributos son atómicos.

Está en 2FN, ya que todos los atributos dependen funcionalmente de manera total de la clave.

Está en 3FN, ya que lo que encontramos a la izquierda de la dependencia funcional es superclave.

Creación de roles y usuarios

Explicación de los permisos de cada rol/usuario. La creación y configuración de dichos permisos se podrá hacer de dos formas:

- 1) A través de código sql. En ese caso se incluirán las sentencias necesarias para crear los roles/usuarios y para establecer sus correspondientes permisos
- 2) A través de la interfaz gráfica de Pgadmin. En ese caso se incluirán capturas de pantalla y explicaciones de los pasos seguidos para crear los roles/usuarios y para establecer sus correspondientes permisos

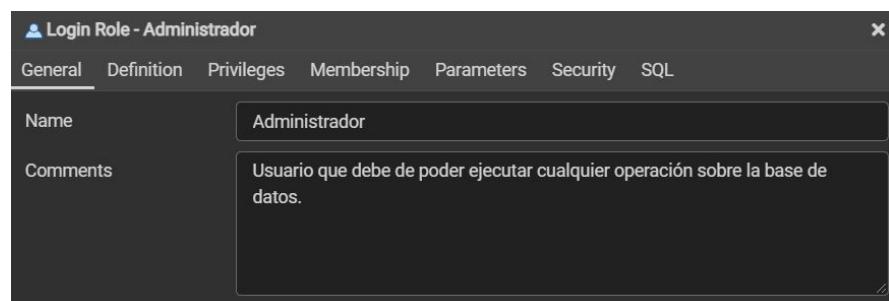
A continuación, se incluirán las capturas de pantalla que verifiquen el funcionamiento establecido en los permisos para cada uno de los usuarios (un ejemplo de operación permitida y otro de no permitida para cada uno de ellos). Para ello será necesario desconectarse y conectarse a la base de datos.

Hemos utilizado la segunda forma para crear los roles. Aquí adjuntamos las explicaciones correspondientes;

1. Rol “Administrador”:

El rol “Administrador” debe poder ejecutar cualquier operación sobre la base de datos, el usuario.

Primero rellenamos los campos de nombre y comentario, para explicar el cometido de ese rol.



Después, marcamos las casillas para conceder los privilegios correspondientes al Administrador. En este caso, al ser un administrador, requiere tener todos los privilegios.E

Login Role - Administrador							
	General	Definition	Privileges	Membership	Parameters	Security	SQL
Can login?			<input checked="" type="checkbox"/> Yes				
Superuser?			<input checked="" type="checkbox"/> Yes				
Create roles?			<input checked="" type="checkbox"/> Yes				
Create databases?			<input checked="" type="checkbox"/> Yes				
Update catalog?			<input checked="" type="checkbox"/> Yes				
Inherit rights from the parent roles?			<input checked="" type="checkbox"/> Yes				
Can initiate streaming replication and backups?			<input checked="" type="checkbox"/> Yes				

El administrador tiene el privilegio de poder revocar privilegios a los demás usuarios.

```
Query Editor Query History Explain Notifications
1 ALTER DEFAULT PRIVILEGES
2   REVOKE ALL ON TABLES FROM "Administrador2";
3 ALTER DEFAULT PRIVILEGES
4   REVOKE ALL ON TABLES FROM "Gestor2";
5 ALTER DEFAULT PRIVILEGES
6   REVOKE ALL ON TABLES FROM "Repcionista2";
```

Data Output Messages

```
ALTER DEFAULT PRIVILEGES
```

Query returned successfully in 160 msec.

Comprobación de permisos con Administrador: INSERT.

```
Query Editor Query History
1 INSERT INTO public."Agente de casting"
2   ("numEmpleado", "Nombre", "DNI", "Código_casting_Casting_presencial", "Calle", "Ciudad", piso, "Portal")
3   VALUES (65, 'Pepe', '07618435T', 0, 'Calle Vargas Llosa', 'Menorca', '5ºE', '3');
```

Data Output Explain Messages Notifications

```
INSERT 0 1
```

Query returned successfully in 49 msec.

Comprobación de permisos con Administrador: UPDATE.

```
Query Editor Query History
1 UPDATE public."Agente de casting"
2   SET "Nombre"='Jose', "DNI"='19706598Y', "Código_casting_Casting_presencial"=0, "Calle"='Calle Rivera',
3   "Ciudad"='Albacete', piso='4ºA', "Portal"='5'
4 WHERE "numEmpleado" = 65
```

Data Output Explain Messages Notifications

```
UPDATE 1
```

Query returned successfully in 55 msec.

Comprobación de permisos con Administrador: CREATE TABLE.

Foreign Data Wrappers
Languages
Schemas (1)
public
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Procedures
Sequences
Tables (21)
Agente de casting
Candidato_adulto
Candidato_adulto_realiza_Prueba_individual
Candidato_nino
Candidato_nino_realiza_Prueba_Individual
Caracol
Casting_online
Casting_online_necesita_Perfil
Casting_presencial
Casting_presencial_necesita_Perfil
Cliente
Cliente_Contrata_Candidato_adulto
Cliente_Contrata_Candidato_nino

PECL3/Administrador@Server

No limit

Query Editor

```
CREATE TABLE public."Caracol"
(
    "Nº_bastidor" bigint NOT NULL,
    PRIMARY KEY ("Nº_bastidor")
);
ALTER TABLE public."Caracol"
    OWNER to "Administrador";
```

Messages

```
ALTER TABLE
Query returned successfully in 59 msec.
```

Comprobación de permisos con Administrador: SELECT.

The screenshot shows a PostgreSQL Query Editor interface. The query entered is:

```
1 SELECT "numEmpleado", "Nombre", "DNI", "Código_casting_Casting_presencial", "Calle", "Ciudad", piso, "Portal"
2 FROM public."Agente de casting";
```

The results are displayed in a table titled "Data Output". The columns are:

	numEmpleado [PK] integer	Nombre character varying (40)	DNI character (9)	Código_casting_Casting_presencial Integer	Calle character varying (20)	Ciudad character varying (20)	piso character
1	10	Federico	02030488A		0 Bernarda	Teruel	4ºF
2	0	Fran	37613746T		1 Calle Unamuno	Barcelona	4ºA
3	1	Antonia	11223344T		1 Calle Miguel Delibes	Madrid	3ºY
4	2	Marta	12345678A		3 Calle Elon Musk	Badalona	5ºH
5	3	Martina	87654321J		4 Calle Violeta	Alicante	9ºD
6	4	Marcos	98765432G		5 Calle Ramón Linares	Alcalá de Henares	2ºR
7	5	Antonio	88776655M		6 Calle Fina Palomares	Guadalajara	6ºG
8	7	Carlos	23465467A		8 Calle Montepinar	Guisconsin	7ºD
9	8	Marta	23657895J		9 Calle Missisipi	Alcantarilla	5ºE
10	9	Amador	41546548J		2 Calle Niu LLork	Chueka	90ºA
11	6	Carla	23579125F		7 Calle Enrique Pastor	Coslada	8ºA
12	11	Jacinta	00000001A		11 Calle Missisipi	Aranzueque	1ºB
13	12	Benigno	99994050J		10 Calle Franciscana	Madrid	80ºA
14	65	Jose	19706598Y		0 Calle Rivera	Albacete	4ºA

Comprobación de permisos con Administrador: DELETE.

The screenshot shows a PostgreSQL Query Editor interface. The query entered is:

```
1 DELETE FROM public."Agente de casting"
2 WHERE "numEmpleado" = 65
```

The results are displayed in a table titled "Data Output". The message shown is:

DELETE 1

Query returned successfully in 50 msec.

2. Rol “Gestor”:

Debe poder manejar los datos de la base de datos (inserción, actualización, borrado y consulta), pero no debe de poder crear nuevas tablas ni elementos que afecten a la estructura de la base de datos.

Primero rellenamos los campos de nombre y comentario, para explicar el cometido de ese rol.

Name	Gestor
Comments	El usuario debe de poder manejar los datos de la base de datos (inserción, actualización, borrado y consulta), pero no debe de poder crear nuevas tablas ni elementos que afecten a la estructura de la base de datos.

Después, marcamos las casillas para conceder los privilegios correspondientes al Gestor. En este caso, debemos asignar los privilegios manualmente en todas las tablas.

Can login?	<input checked="" type="checkbox"/> Yes
Superuser?	<input type="checkbox"/> No
Create roles?	<input type="checkbox"/> No
Create databases?	<input type="checkbox"/> No
Update catalog?	<input type="checkbox"/> No
Inherit rights from the parent roles?	<input type="checkbox"/> No
Can initiate streaming replication and backups?	<input type="checkbox"/> No

Dado que el usuario “gestor” solo tiene los permisos de selección, inserción, borrado y actualización (y aplicando la interfaz gráfica de pgAdmin 4 estos solo se pueden asignar tabla por tabla), realizaremos esta misma acción (imagen inferior) con todas las tablas de nuestra BBDD, en este caso empezaremos por conceder los permisos ya mencionados al usuario gestor para la tabla “TelefonoRepresentante”.

<input type="checkbox"/> ALL	<input type="checkbox"/> WITH GRANT OPTION
<input checked="" type="checkbox"/> SELECT	<input type="checkbox"/> WITH GRANT OPTION
<input checked="" type="checkbox"/> INSERT	<input type="checkbox"/> WITH GRANT OPTION
<input checked="" type="checkbox"/> DELETE	<input type="checkbox"/> WITH GRANT OPTION
<input checked="" type="checkbox"/> UPDATE	<input type="checkbox"/> WITH GRANT OPTION
<input type="checkbox"/> TRUNCATE	<input type="checkbox"/> WITH GRANT OPTION
<input type="checkbox"/> REFERENCES	<input type="checkbox"/> WITH GRANT OPTION
<input type="checkbox"/> TRIGGER	<input type="checkbox"/> WITH GRANT OPTION

Comprobación de permisos con Gestor: INSERT.

```
PECL3/Gestor@PostgreSQL 13
Query Editor Query History
1 INSERT INTO public."Agente de casting"(
2   "numEmpleado", "Nombre", "DNI", "Código_casting_Casting_presencial", "Calle", "Ciudad", piso, "Portal")
3   VALUES (65, 'Paco', '94318169P', 0, 'Calle calle', 'Madrid', '3ºD', '3');
```

Data Output Explain Messages Notifications

INSERT 1

Query returned successfully in 58 msec.

Comprobación de permisos con Gestor: UPDATE.

```
Query Editor Query History
1 UPDATE public."Agente de casting"
2   SET "Nombre"='Juanjo', "DNI"='94670489H', "Código_casting_Casting_presencial"=1, "Calle"='Miguel Maldonado',
3   "Ciudad"='Malaga', piso='6ºB', "Portal"='6'
4   WHERE "numEmpleado"=65
```

Data Output Explain Messages Notifications

UPDATE 1

Query returned successfully in 53 msec.

Comprobación de permisos con Gestor: DELETE.

```
Query Editor Query History
1 DELETE FROM public."Agente de casting"
2 WHERE "numEmpleado"=65
```

Data Output Explain Messages Notifications

DELETE 1

Query returned successfully in 53 msec.

Comprobación de permisos con Gestor: SELECT.

```
Query Editor Query History
1 SELECT "numEmpleado", "Nombre", "DNI", "Código_casting_Casting_presencial", "Calle", "Ciudad", piso, "Portal"
2   FROM public."Agente de casting";
```

Data Output Explain Messages Notifications

	numEmpleado [PK] integer	Nombre character varying (40)	DNI character (9)	Código_casting_Casting_presencial integer	Calle character varying (20)	Ciudad character varying (20)	piso character
1	10	Federico	02030488A	0	Bernardo	Teruel	4%
2	0	Fran	37613746T	1	Calle Unamuno	Barcelona	4%
3	1	Antonia	11223344*	1	Calle Miguel Delibes	Madrid	3%
4	2	Marta	12345678A	3	Calle Elon Musk	Badalona	5%
5	3	Martina	87654321J	4	Calle Violeta	Alicante	9%
6	4	Marcos	98765432G	5	Calle Ramón Linares	Alcalá de Henares	2%
7	5	Antonio	88776655M	6	Calle Fina Palomares	Guadalajara	6%
8	7	Carlos	23465467A	8	Calle Montepinar	Guisconsin	7%
9	8	Marta	23657895J	9	Calle Missisipi	Alcantarilla	5%
10	9	Amador	41546548J	2	Calle Niu LLork	Chueka	90%
11	6	Carla	23579125F	7	Calle Enrique Pastor	Coslada	8%
12	11	Jacinta	00000001A	11	Calle Missisipi	Aranzueque	1%
13	12	Benigno	99994050J	10	Calle Franciscana	Madrid	80%

Comprobación de permisos con Gestor: CREATE TABLE.

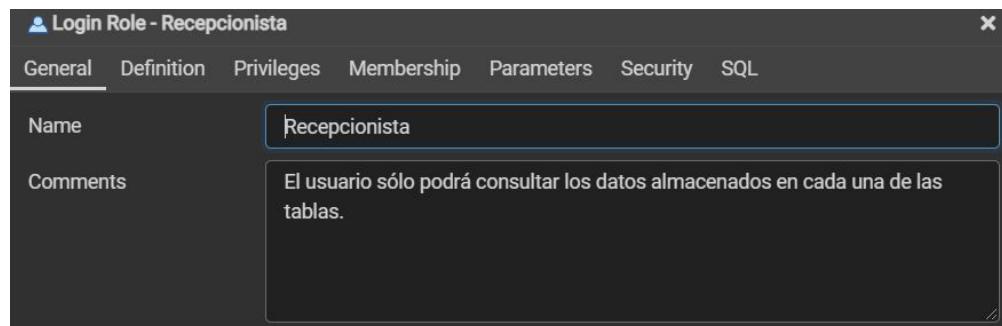
The screenshot shows the pgAdmin interface. In the top right, there's a 'Messages' panel with an error message: "ERROR: permiso denegado al esquema public LINE 1: CREATE TABLE public.\"Caracol\"". Below it, the SQL state is listed as 42581 and the character position as 14. In the bottom left, a 'Server' configuration dialog is open, showing a 'Connection' tab with 'Maintenance database' set to 'postgres', 'Username' set to 'Gestor', and other fields like 'Role' and 'Service' empty. Buttons for 'Cancel', 'Reset', and 'Save' are at the bottom.

```
1 CREATE TABLE public.\"Caracol\"  
2 (  
3     \"Nº_bastidor\" bigint NOT NULL,  
4     PRIMARY KEY (\"Nº_bastidor\")  
5 );  
6  
7 ALTER TABLE public.\"Caracol\"  
8     OWNER to \"Gestor\";
```

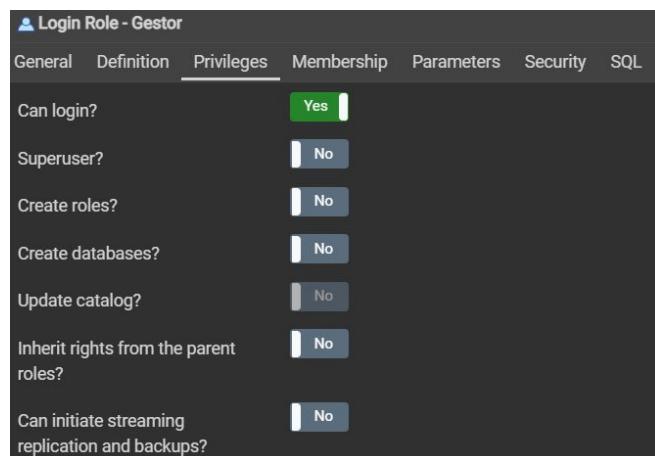
3. Rol “Repcionista”:

Sólo podrá consultar los datos almacenados en cada una de las tablas.

Primero rellenamos los campos de nombre y comentario, para explicar el cometido de ese rol.



Después, marcamos las casillas para conceder los privilegios correspondientes al Repcionista. En este caso, debemos asignar los privilegios manualmente en todas las tablas.



Dado que el usuario recepcionista solo tiene los permisos de selección (y aplicando la interfaz gráfica de pgAdmin 4 estos solo se pueden asignar tabla por tabla), realizaremos esta misma acción (imagen inferior) con todas las tablas de nuestra BBDD, en este caso empezaremos por conceder permisos al usuario “representante” para realizar operaciones de “consulta” a la tabla “TelefonoRepresentante”.

User	Role	Permissions
Gestor	radw	ALL, SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER
Recepcionista	radw	SELECT

Comprobación de permisos con Recepcionista: SELECT.

```

SELECT "numEmpleado" FROM "Agente de casting"

```

numEmpleado
1
2
3
4
5
6
7
8
9
10
11
12
13

Comprobación de permisos con Recepcionista: INSERT.

```

INSERT INTO public."Agente de casting"(
    "numEmpleado", "Nombre", "DNI", "Código_casting_Casting_presencial", "Calle", "Ciudad", piso, "Portal")
VALUES (20, 'Pepe', '28135795G', 0, 'Ferraz', 'Moscú', '6ºB', 'A');

```

ERROR: permiso denegado a la tabla Agente de casting
SQL state: 42501

Comprobación de permisos con Recepcionista: DELETE.

The screenshot shows the pgAdmin Query Editor interface. The title bar says "PECL3/Recepcionista@PostgreSQL 13". The main area contains the SQL command:

```
1 DELETE FROM public."Agente de casting"
```

Below the command, the "Messages" tab is selected, showing the error message:

```
ERROR: permiso denegado a la tabla Agente de casting  
SQL state: 42501
```

Comprobación de permisos con recepcionista: UPDATE.

The screenshot shows the pgAdmin Query Editor interface. The title bar says "PECL3/Recepcionista@PostgreSQL 13". The main area contains the SQL command:

```
161 UPDATE public."Agente de casting"  
162     SET "Nombre"= 'Pedro', "DNI"='94670481I', "Código_casting_Casting_presencial"=1, "Calle"='Calle Maldonado',  
163     "Ciudad"='Malaga', piso='6ºB', "Portal"='6'  
164 WHERE "numEmpleado"= 0;  
165  
166
```

Below the command, the "Messages" tab is selected, showing the error message:

```
ERROR: permiso denegado a la tabla Agente de casting  
SQL state: 42501
```

Comprobación de permisos con Recepcionista: CREATE TABLE.

The screenshot shows the pgAdmin Query Editor interface. The title bar says "PECL3/Recepcionista@PostgreSQL 13". The main area contains the SQL command:

```
1 CREATE TABLE public."Caracol"  
2 (  
3     "Nº_bastidor" bigint NOT NULL,  
4     PRIMARY KEY ("Nº_bastidor")  
5 );  
6  
7 ALTER TABLE public."Caracol"  
8     OWNER to "Recepcionista";
```

To the right of the query editor, a "Messages" window is open, displaying the error message:

```
ERROR: permiso denegado al esquema public  
LINE 1: CREATE TABLE public."Caracol"  
^  
SQL state: 42501  
Character: 14
```

Below the messages window, a "Server" configuration dialog is visible. It has tabs for "General", "Connection", "SSL", "SSH Tunnel", and "Advanced". The "Connection" tab is selected, showing the following settings:

Host name/address	localhost
Port	5432
Maintenance database	postgres
Username	Recepcionista

Buttons at the bottom of the dialog include "Cancel", "Reset", and "Save".

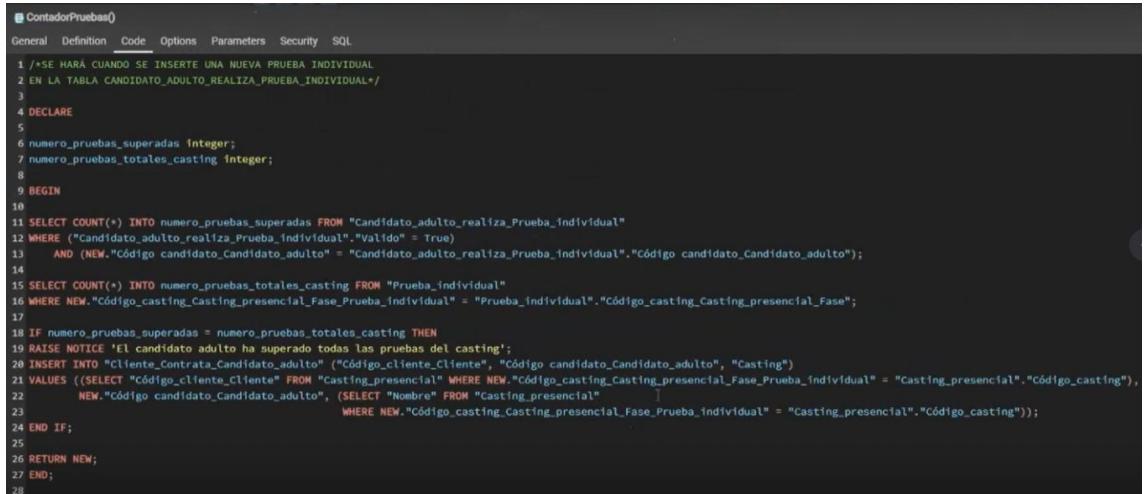
Implementación de disparadores

Se indicará el funcionamiento de cada disparador y cuándo se ejecuta. Para cada uno de ellos se incluirá el código de la función disparadora y una captura con la configuración del disparador. A continuación, se ejecutará la sentencia que disparen dicho código y se incluirán capturas de pantalla en las que se verifique el correcto funcionamiento del disparador.

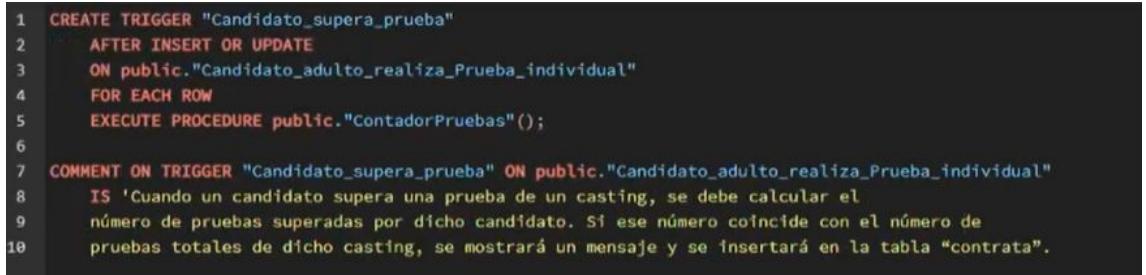
-Calcular el importe total que ha pagado cada candidato teniendo en cuenta todas las pruebas que ha realizado.

```
1 BEGIN
2
3 UPDATE "Candidato_adulto"
4   SET "Coste_total" = (SELECT SUM("Coste") FROM "Prueba_individual", "Candidato_adulto_realiza_Prueba_individual" WHERE "Candidato_adulto_realiza_Prueba_individual"."Código_precio" = NEW."Código_candidato_Candidato_adulto");
5
6
7 RAISE NOTICE 'Se ha actualizado el importe total a pagar por el candidato %', NEW."Código_candidato_Candidato_adulto"; -- No muestro el nombre del candidato porque puede haber
8
9 RETURN NULL;
10 END;
11
```

-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”.



```
ContadorPruebas()
General Definition Code Options Parameters Security SQL
1 /*SE HARÁ CUANDO SE INSERTE UNA NUEVA PRUEBA INDIVIDUAL
2 EN LA TABLA CANDIDATO_ADULTO_REALIZA_PRUEBA_INDIVIDUAL*/
3
4 DECLARE
5
6 numero_pruebas_superadas integer;
7 numero_pruebas_totales_casting integer;
8
9 BEGIN
10
11 SELECT COUNT(*) INTO numero_pruebas_superadas FROM "Candidato_adulto_realiza_Prueba_individual"
12 WHERE ("Candidato_adulto_realiza_Prueba_individual"."Valido" = True)
13 AND (NEW."Código_candidato_Candidato_adulto" = "Candidato_adulto_realiza_Prueba_individual"."Código_candidato_Candidato_adulto");
14
15 SELECT COUNT(*) INTO numero_pruebas_totales_casting FROM "Prueba_individual"
16 WHERE NEW."Código_casting_Casting_presencial_Fase_Prueba_Individual" = "Prueba_Individual"."Código_casting_Casting_presencial_Fase";
17
18 IF numero_pruebas_superadas = numero_pruebas_totales_casting THEN
19 RAISE NOTICE 'El candidato adulto ha superado todas las pruebas del casting';
20 INSERT INTO "Cliente_Contrata_Candidato_adulto" ("Código_cliente_Cliente", "Código_candidato_Candidato_adulto", "Casting")
21 VALUES ((SELECT "Código_cliente_Cliente" FROM "Casting_presencial" WHERE NEW."Código_casting_Casting_presencial_Fase_Prueba_Individual" = "Casting_presencial"."Código_casting"),
22          NEW."Código_candidato_Candidato_adulto", (SELECT "Nombre" FROM "Casting_presencial"
23          WHERE NEW."Código_casting_Casting_presencial_Fase_Prueba_Individual" = "Casting_presencial"."Código_casting"));
24 END IF;
25
26 RETURN NEW;
27 END;
28
```



```
CREATE TRIGGER "Candidato_supera_prueba"
AFTER INSERT OR UPDATE
ON public."Candidato_adulto_realiza_Prueba_individual"
FOR EACH ROW
EXECUTE PROCEDURE public."ContadorPruebas"();

COMMENT ON TRIGGER "Candidato_supera_prueba" ON public."Candidato_adulto_realiza_Prueba_individual"
IS '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".'
```

Query Editor Query History Explain Notifications

```

1 INSERT INTO public."Candidato_adulto_realiza_Prueba_individual"(
2   "Código_prueba_Prueba_individual", "Código_de_fase_Fase_Prueba_individual", "Código_casting_Casting_presencial_F
3   VALUES
4   (10, 10, 9, 2002, True); |

```

Data Output Messages

NOTICE: El candidato adulto ha superado todas las pruebas del casting
INSERT 0 1

Query returned successfully in 380 msec.

Data Output Messages

Código_cliente_Cliente [PK] integer	Código_candidato_Candidato_adulto [PK] integer	Casting character (40)
1	0	2006 C.Online C.Tangana...
2	1	2000 C.Rosas --
3	2	2003 C.Presencial C.Pay...
4	4	2000 C.Presencial C.Ant...
5	4	2001 C.Presencial C.Ant...
6	6	2002 C.Cuarto Milenio --
7	6	2004 C.Online C.Videojue...

-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.

```

1 DECLARE
2
3 candidatos_adultos_contratados_para_un_mismo_casting integer;
4
5 numero_personas_necesita_casting integer;
6
7 BEGIN
8
9 candidatos_adultos_contratados_para_un_mismo_casting = (SELECT COUNT ("Código_candidato_Candidato_adulto") FROM "Cliente_Contrata_Candidato_adulto" WHERE "Cliente_Contrata_Can
10
11 numero_personas_necesita_casting = (SELECT "NumPersonas" FROM "Casting_presencial" WHERE NEW."Casting" = "Casting_presencial"."Código_casting");
12
13 IF candidatos_adultos_contratados_para_un_mismo_casting = numero_personas_necesita_casting THEN
14 RAISE NOTICE 'Se acaba de contratar a todas las personas del casting.';
15
16 ELSEIF candidatos_adultos_contratados_para_un_mismo_casting < numero_personas_necesita_casting THEN
17 RAISE NOTICE 'Se han contratado % personas de % personas que tiene el casting.',candidatos_adultos_contratados_para_un_mismo_casting, numero_personas_necesita_casting;
18
19 ELSE
20 RAISE EXCEPTION 'SE HA PRODUCIDO UN ERROR: NO SE PUEDE CONTRATAR A MÁS PERSONAS DE LAS QUE HAY EN UN CASTING。';
21
22 END IF;
23 RETURN NULL;
24 END;

```

```

1 INSERT INTO public."Cliente_Contrata_Candidato_adulto"(  

2 "Código_cliente_Cliente", "Código_candidato_Candidato_adulto", "Casting")  

3 VALUES (0, 2005, 10);

Data Output Messages

NOTICE: Se han contratado 2 personas de 20 personas que tiene el casting.
INSERT 0 1

Query returned successfully in 593 msec.

```

Query Editor		Query History		Explain	Notifications	Data Output	Messages
	Código_cliente_Cliente [PK] integer		Código_candidato_Candidato_adulto [PK] integer			Casting integer	
1		0			2005	10	
2		0			2006	2003	
3		1			2000	0	
4		2			2003	2002	
5		4			2000	2001	
6		4			2001	10	
7		6			2002	9	
8		6			2004	2000	

-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

```

Dashboard Properties SQL Statistics Dependencies Dependents PECL2_BBDD/p... ConsultasPECL... PECL2_BBDD/p... TRIGERS_PECL3.sql
PECL2_BBDD/postgres@Servidor PECL3
Query Editor Query History Explain Notifications
73 /*  

74  

75 SE HARÁ CUANDO SE INSERTE O MODIFIQUE UNA NUEVA PRUEBA INDIVIDUAL (Esta última condición la añadimos para cuando se produzca un error por parte de los calificadores,  

76 de manera que den por válida una prueba a un candidato que previamente no la había superado, de esta manera se podrán subsanar esos errores y el trigger podrá saltar igual  

77 EN LA TABLA CANDIDATO_ADULTO_REALIZA_PRUEBA_INDIVIDUAL  

78  

79 */  

80  

81 /* BEGIN  

82  

83 IF (SELECT "Codigo_perfil_Perfil" FROM "Candidato_adulto"  

84 WHERE NEW."Código_candidato_Candidato_adulto" = "Candidato_adulto"."Código_candidato") = (SELECT "Codigo_perfil_Perfil" FROM "Casting_presencial_necesita_Perfil"  

85 WHERE NEW."Código_casting_Casting_presencial_Fase_Probea_Individual" = "Casting_presencial_necesita_Perfil."  

86 THEN  

87 RAISE NOTICE 'El Candidato que acaba de hacer la prueba tiene el perfil que necesita el casting que las propone.';  

88 ELSE  

89 RAISE NOTICE 'El Candidato que acaba de hacer la prueba NO tiene el perfil que necesita el casting que las propone.';  

90  

91 END IF;  

92 END;  

93  

94

```

```

PECL2_BBDD/postgres@Servidor PECL3
Query Editor  Query History  Explain  Notifications
1 INSERT INTO public."Candidato_adulto_realiza_Prueba_individual"
2   "Código_prueba_Prueba_Individual", "Código_de_fase_Fase_Prueba_Individual", "Código_casting_Casting_presencial_Fase_Prueba_individual", "Código_candidato_Candida
3 VALUES
4   (3, 3, 2, 2008, True);

Data Output  Messages
NOTICE: El candidato adulto ha superado todas las pruebas del casting
NOTICE: El Candidato que acaba de hacer la prueba tiene el perfil que necesita el casting que las propone.

ERROR: la ejecución alcanzó el fin del procedimiento disparador sin encontrar RETURN
CONTEXT: función PL/pgSQL "Comprobar_perfil"
SQL state: 2F005

```

```

8
9
10 INSERT INTO public."Candidato_adulto_realiza_Prueba_individual"
11   "Código_prueba_Prueba_Individual", "Código_de_fase_Fase_Prueba_Individual", "Código_casting_Casting_presencial_Fase_Prueba_individual", "Código_candidato_Candida
12 VALUES
13   (9, 9, 8, 2008, True);

Data Output  Messages
NOTICE: El candidato adulto ha superado todas las pruebas del casting
NOTICE: El Candidato que acaba de hacer la prueba NO tiene el perfil que necesita el casting que las propone.

ERROR: la ejecución alcanzó el fin del procedimiento disparador sin encontrar RETURN
CONTEXT: función PL/pgSQL "Comprobar_perfil"
SQL state: 2F005

```

Acceso desde Java

Se mostrarán capturas del programa implementado: una pequeña interfaz gráfica en la que se pueda acceder a la base de datos con los distintos usuarios y se puedan ejecutar las consultas en sql. Se incluirá un directorio que incluya el proyecto Java.

Nota: Se incluirá un fichero backup de la base de datos para que los profesores puedan recrear la base de datos en pgadmin.

Antes de nada aclarar que las consultas 4 y 18 no se han realizado. La consulta 4 es debido a un mal diseño de la base de datos. La 18 no supimos cómo obtener exactamente lo que se pedía. Sin embargo, el resultado similar tiene una query y está comentada en el código.

Primero comenzaremos explicando la ejecución básica del programa, el *main*.

```

public static void main(String args[]) throws InstantiationException, IllegalAccessException {
    String entradaTeclado2;
    Scanner entradaEscaner = new Scanner(System.in);
    boolean condition = false;
    while (!condition) {
        Conector conector = new Conector();
        conector.connect();

        conector.q1();
        conector.q2();
        conector.q3();
        conector.q5();
        conector.q6();
        conector.q7();
        conector.q8();
        conector.q9();
        conector.q10();
        conector.q11();
        conector.q12();
        conector.q13();
        conector.q14();
        conector.q15();
        conector.q16();
        conector.q17();
                conector.q18();
        conector.q19();
        conector.q20();

        System.out.println("Si desea salir, introduzca 0");
        entradaTeclado2 = entradaEscaner.nextLine();
        if (entradaTeclado2.equals("0")) {
            condition = true;
        }
        conector.disconnect();
    }
}

```

Primero, se llama al método `connect()`, donde se conectará con el servidor con el usuario que deseemos.

A continuación, se nos mostrará el resultado de la ejecución de todas las consultas.

Al acabar con la última, el programa nos solicita que le indiquemos si deseamos volver a ejecutarlo. Esto es para poder probarlo con otros usuarios. El programa estará en bucle hasta que introduzcamos '0' al final de una ejecución.

Si se introduce '0', se procede al método `disconnect()`, con el que nos desconectaremos de la base de datos y se acabará la ejecución del programa.

Ahora hablaré sobre los distintos métodos:

- **`Connect()`**

```
public Connection connect() {
    System.out.println("Antes de establecer la conexión, debe elegir el usuario:");
    System.out.println("Administrador = 1, Gestor = 2, Recepcionista = 3");

    while (!correcto) {

        entradaTeclado = entradaEscaner.nextLine();
        //entradaTeclado = "1"; //Esto es para asignar por fuerza el valor de Administrador.

        if (entradaTeclado.equals("1")) {
            user = "Administrador";
            correcto = true;
        } else if (entradaTeclado.equals("2")) {
            user = "Gestor";
            correcto = true;
        } else if (entradaTeclado.equals("3")) {
            user = "Recepcionista";
            correcto = true;
        } else {
            System.out.println("Introduzca un numero valido");
        }
    }

    try {
        Class.forName("org.postgresql.Driver").newInstance();
        conn = DriverManager.getConnection(url, user, password);
        System.out.println("Conectado!!!");
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    } catch (ClassNotFoundException | InstantiationException | IllegalAccessException ex) {
        System.out.println("Driver no conectado");
    }
    return conn;
}
```

Este metodo está dividido en 2 fases.

La primera fase sirve como “interfaz de selección” para el usuario. Dependiendo de lo que introduzca, entrará a la BBDD con un usuario determinado, o no entrará.

La segunda fase sirve para conectarse con la BBDD. Los métodos, son los mostrados en las diapositivas de clases.

- ***Disconnect()***

```
public void disconnect() {
    try {
        conn.close();
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
}
```

Este es un sencillo método que sirve para desconectar el programa de la BBDD. Como he citado anteriormente, se usa al final de la ejecución del programa en caso de que el usuario desee salir.

- ***qn()***

Realmente no hay ningún método llamado *qn()*. La ‘n’ hace referencia a todos los métodos existentes desde el método *q1()* hasta el *q20()*.

Cada uno de los métodos es una consulta distinta. Aquí un ejemplo de una de las consultas más complejas.

```
public void q15() {
    try {
        System.out.println("-----");
        + "\n" + "----- Decimoquinta consulta -----";
        + "\n" + "-----");
        System.out.println("MOSTRAR EL DINERO TOTAL RECAUDADO POR LA EMPRESA" + "\n");
        System.out.println("Dinero_total");

        String query1 = "CREATE VIEW \"Pagos_candidatos\" AS\n"
            + "SELECT \"Código_candidato_Candidato_adulto\", T1.\"Coste\"\n"
            + "      FROM (\"Candidato_adulto_realiza_Prueba_individual\" INNER JOIN \"Prueba_individual\" \n"
            + "      ON \"Candidato_adulto_realiza_Prueba_individual\".\"Código_prueba_Prueba_individual\" = \"Prueba_individual\".\"Código_prueba\") as T1\n"
            + "UNION SELECT \"Código_candidato_Candidato_ninio\", T1.\"Coste\" as Coste_total\n"
            + "      FROM (\"Candidato_ninio_realiza_Prueba_individual\" INNER JOIN \"Prueba_individual\" \n"
            + "      ON \"Candidato_ninio_realiza_Prueba_individual\".\"Código_prueba_Prueba_individual\" = \"Prueba_individual\".\"Código_prueba\") as T1\n"
            + "          ORDER BY \"Código_candidato_Candidato_adulto\"";
        Statement st = conn.createStatement();
        st.executeUpdate(query1);

        String query2 = "CREATE VIEW \"Pagos_clientes\" AS\n"
            + "SELECT \"Código_cliente_Cliente\", \"Coste\"\n"
            + "      FROM \"casting_online\"\n"
            + "UNION SELECT \"Código_cliente_Cliente\", \"Coste\"\n"
            + "      FROM \"Casting_presencial\"";
        st.executeUpdate(query2);

        String query3 = "CREATE VIEW \"COSTE_TOTAL_CLIENTES_CANDIDATOS\" AS\n"
            + "SELECT SUM(\"Coste\") FROM \"Pagos_clientes\" UNION SELECT SUM(\"Coste\") FROM \"Pagos_candidatos\"";
        st.executeUpdate(query3);

        String query4 = "SELECT SUM(\"sum\") FROM \"COSTE_TOTAL_CLIENTES_CANDIDATOS\"";
        ResultSet rs = st.executeQuery(query4);
```

```

        while (rs.next()) {
            System.out.println(rs.getString(1) + "\n");
        }

        String query5 = "DROP VIEW \"COSTE_TOTAL_CLIENTES_CANDIDATOS\"";
        st.executeUpdate(query5);

        String query6 = "DROP VIEW \"Pagos_candidatos\"";
        st.executeUpdate(query6);

        String query7 = "DROP VIEW \"Pagos_clientes\"";
        st.executeUpdate(query7);

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

```

Nota: El código no cabe en una sola captura. Después de ejecutar la query4, es cuando entra en el bucle while, a la misma altura.

Como podemos observar, tenemos una estructura ciertamente repetitiva, en todos los métodos *qn()*:

1. Primero vamos a encontrar siempre las vistas, serán mandadas a ejecutar las primeras.
2. Despues, se pasarán las consultas reales, y se almacenarán en un *resultset*, para poder obtener el resultado de la ejecución de las mismas.
3. Estos resultados se sacarán del *resultset* en un bucle *while*, y se mostrarán por pantalla.

Las vistas se deben ejecutar de una en una, en ‘consultas’ distintas. Y se debe usar el método “*.executeUpdate(query)*” de la clase *Statement*. Este método nos permite ejecutar y refrescar la base de datos.

Después de mostrar los resultados de la consulta, hemos optado por borrar las vistas creadas anteriormente. Esto se debe a que si se intenta ejecutar una consulta de una creación de una vista existente, nos saltará un error. Por ello, detrás de los *while* de consultas con vistas, habrá nuevas consultas con “*drop*”. Estas consultas son exactamente igual a la hora de ejecutar que las de las vistas, de una en una en consultas distintas y con “*.executeUpdate(query)*”.

El resto de elementos que nos podemos encontrar en las consultas son *prints* introducidos para dar una estética al programa.

Documentación detallada.

Problema 1: Comenzamos corrigiendo el pgmodeler de la práctica anterior agregando las herencias. El problema surgió a la hora de trabajar con esas herencias en pgadmin ya que no supimos cómo funcionaban en ese sistema gestor. Por esto hemos continuado con nuestro diagrama del modelo relacional sin modificar (perteneciente a la PECL 2). Pero como se ha podido apreciar en el apartado de la normalización, normalizamos tanto el diagrama que consta de herencias con la herramienta de pgmodeler como las entidades pertenecientes al diagrama sin herencias (que fue el que finalmente decidimos implementar en pgAdmin para ganar tiempo, a pesar de los problemas que nos ha acarreado esta decisión).

Problema 2: Al realizar consultas con vistas. En un primer momento, no sabíamos cómo implementarlas en Java. Realizamos distintas pruebas hasta que nos dimos cuenta de que para la manera en que queríamos trabajar con ellas, era necesario eliminarlas mediante los *DROP*

VIEWS (en algunos casos *CASCADE*). Dado que las vistas se creaban, pero las consultas hechas en java no las reconocían, caímos en la cuenta de que era necesario refrescar o actualizar el estado de la BBDD. Por ello, las vistas se mandan en `.executeUpdate`, además de que no se pueden mandar 2 o más vistas o drops en una misma query.

Problema 3: Al usar una base de datos sin aplicar herencias de la herramienta pgmodeler (ya que nosotros transformamos nuestras herencias del modelo ER al modelo relacional de forma manual como vimos en teoría), implementar los trigger acarreó una serie de problemas que se tradujeron en una mayor dedicación de tiempo a esta parte de la práctica (ya de por sí la más ardua), ya que en vez de hacer los 4 triggers obligatorios tuvimos que implementar 8. Esto nos hizo caer una vez más en la cuenta de lo beneficioso que es tener una base de datos bien diseñada desde las primeras fases (lo que al principio puede llevar más tiempo, al final supone una importante reducción de carga de trabajo en las últimas fases del desarrollo).

Problema 4: Durante la creación de los roles topamos con el problema de que al trabajar con los roles de Gestor y Recepcionista, a los cuales no les concedimos el derecho de crear tablas de manera implícita mediante la interfaz de PgAdmin, seguían pudiendo crearlas. Esto no habríamos sido capaces de resolverlo sin la tutoría realizada el jueves de esta misma semana.

Por otro lado, comentar la metodología de trabajo utilizada. Para realizar la práctica, hemos utilizado la plataforma *Discord* (videoconferencias), teniendo en cuenta los tiempos que corren.

Además, hemos utilizado como herramienta el sistema controlador de versiones *Github* para organizarnos, y compartir de manera más eficiente los archivos que hemos usado, pudiendo llevar un registro de las modificaciones que se iban realizando en la práctica por cada uno de sus miembros. Aquí una foto que muestra cómo la hemos empleado;

The screenshot shows a GitHub repository interface. At the top, it displays 'main' (branch), '1 branch', '0 tags', 'Go to file', 'Add file', and a green 'Code' button. Below this is a list of commits:

Author	Message	Time Ago
cesarmunuera	añadidos cambios	115789b 16 minutes ago
	Java_BBDD	pequeños cambios
	.gitignore	Creado el proyecto de Java, y a la mitad
	BBDD_PECL3.sql	backup BBDD
	ComprobacionPermisosUsuarios.sql	Consultas relativas a usuarios
	ComprobacionTriggers.sql	añadidos elementos en las comprobaciones de los triggers
	ConsultasPECL2.sql	acabado, faltan detalles
	CreacionUsuariosConcesionPermisos....	Consultas relativas a usuarios
	Diagrama Modelo Relacional PECL2.d...	añadidos cambios
	Diagrama Modelo Relacional PECL3.d...	añadidos cambios
	Enunciado PECL3.pdf	añadidos cambios
	README.md	añadidos cambios
	TRIGGERS_PECL3.sql	Creación de Triggers y sus comprobaciones SQL

Modelo relacional aplicando la herramienta de pgmodeler para las herencias que no usamos.

