1. Normalización de las tablas

**Tabla Casting:**

cod\_casting(PK): A; nombre: B; descripción: C; tipo: D; cod\_cliente(FK): E

A->BCDE E->A

Normalización:

R1: {A,B,C,D}

R2: {A,E}

**Tabla Casting\_online:**

cod\_casting(FK): A; num\_personas: B; fecha: C; web: D;

A->BCD

Ya está en 3FN

**Tabla Casting\_presencial:**

cod\_casting(FK): A; num\_personas: B; fecha: C; cod\_empleado(FK): D;

A->BCD

Ya está en 3FN

R1:{A,B}

**Tabla Casting\_perfil:**

cod\_perfil(PK): A; cod\_casting(FK): B

B -> A

Ya está en 3FN

R1:{A,B}

**Tabla Cliente:**

cod\_cliente(PK): A; nombre: B; dirección: C; telefono: D; contacto: E; actividad: F

Respecto a anteriores entregas vamos a dividir nombre(B) en nombre(B), apellido1(G) y apellido2(H)

Por tanto las df quedarían: A->BCDEFGH y B->GH

Normalización:

R1:{A,C,E,F}

R2:{A,B,G,H}

R3:{A,D] -> por si el cliente proporcionase varios teléfonos

**Tabla Candidato:**

cod\_candidato(PK): A; nombre: B; apellido1: C; apellido2: D; direccion: E; telefono: F; nacimiento: G; importe: H; dni\_representante(FK): I; cod\_perfil(FK): J

A->BCDEFGHIJ

B->CD

Normalización

R1:{A,E,G,H,I,J}

R2:{A,B,C,D}

R3:{A,F} -> por si el cliente proporcionase varios teléfonos

**Tabla Adulto:**

cod\_candidato(PK): A; dni:B

A->B

Ya se encuentra en 3FN

R1:{A,B}

**Tabla Niño:**

cod\_candidato(PK): A; nombre\_tutor:B

A->B

Ya se encuentra en 3FN

R1:{A,B}

**Tabla Perfil:**

cod\_perfil(PK): A; provincia: B; sexo: C; rango\_edad: D; rango\_altura: E; color\_pelo: F; color\_ojos: G; especialidad: H; experiencia: I

A -> BCDEFGHI

Ya se encuentra en 3FN

R1:{A,B,C,D,E,F,G,H,I}

Tabla Pruebas:

cod\_prueba(PK): A; sala: B; descripcion: C; precio:D; cod\_fase(FK): E

A -> BCD E->A

Normalización:

R1:{A,B,C,D}

R2:{A,E}

**Tabla representante:**

dni(PK): A; nombre: B; direccion: C; telefono:D

A -> BCD

Ya se encuentra en 3FN

R1:{A,B,C,D}

**Tabla resultado\_pruebas:**

cod\_prueba(FK):A; cod\_candidato(FK):B; superada: C

A -> BC

Ya se encuentra en 3FN

R1:{A,B,C}

**Tabla fase:**

cod\_fase(PK): A; cod\_casting(FK): B

B->A

Ya se encuentra en 3FN

R1:{A,B}

**Tabla agente\_casting:**

cod\_empleado(PK): A; dni: B; nombre: C; direccion: D

AB -> CD

Ya se encuentra en 3FN

R1:{A,B,C,D}

2.

Creamos los triggers:

El primero actualiza el importe del candidato cuando se inserta una nueva fila en resultado\_prueba añadiéndole al total del candidato el precio de dicha prueba

El segundo añade a la tabla contrata a aquel candidato que haya superado todas las pruebas de un casting, comparando las pruebas superadas del candidato con las totales del casting.

El tercer trigger complementa al segundo, ya que cuando un casting ya ha completado su total de personas contratadas requeridas no acepta más inserts en la tabla contrata.

Por último el trigger final comprueba que un candidato encaja en alguno de los perfiles que el casting al que se apunta contiene.

3.

Para poder realizar la conexión a la base de datos, debimos descargar de la aplicaión Stack Builder el jdbc necesario para poder unir java y postgres. Posteriormente, debimos incluir el .jar que contenía el jdbc en java. Después de esto tuvimos un problema ya que al crear un nuevo proyecto no nos dejaba incluir dicho .jar en los archivos del proyeto, y para poder solucionarlo, tuvimos que cambiar el archivo pom.xml para que pudieramos incluir el .jar en las dependencias.

Por último incluímos el archivo .java que nos facilitaste con el que pudimos realizar la conexión con la base de datos.

4.

Para la creación de usuarios, lo único que tuvimos que hacer fue crear los respectivos usuarios con las contraseñas que hacen referencia a su propio nombre y les asignamos los privilegios pertinente. Al administrador se le dió el permiso de poder hacer cualquier operación en la base de datos "Casting". Al gestor se le dieron los privilegios de incluir,borrar,actualizar o consultar información en todas las tablas.

Y, por último, el recepcionista simplemente es capaz de consultar la información en todas las tablas.

5.

La modificación que hicimos en el archivo Postgres.java fue simplemente la capacidad de poder iniciar sesión a través de la consola del propio compilador, en este caso Netbeans.

Para ello, creamos 3 variables globales(usuario,contraseña y nombreBaseDatos) con las que mediante la inclusión de la herramienta Scanner pudimos almacenar los datos introducidos por el teclado en dichas variables y posteriormente incluirlas en la función DriverManager.getConnection() y así seremos capaces de iniciar sesión con el usuario que se prefiera.

Para finalizar, también incluimos la posibilidad de escribir por teclado la operación que se quiera realizar en la base de datos.