Grado en Ingeniería Informática
Universidad de Granada

Práctica 3: Diseño de datos en un Sistema de Información

Sistema de Gestión de un Centro Deportivo

Autores: Grupo A1 (NoQueremosPerderLaBeca)

- Heredia Muñoz, José
- Hernandez Aranguren, Pedro
- Pérez Meneses, Francisco Alejandro
- Salazar Cano, Santiago

Índice

1. Descripción del Sistema

El Sistema de Gestión de Centros Deportivos (SGCD) es un Sistema de Información destinado a la administración y procesamiento de datos necesarios para el tipo de negocio que es un centro deportivo con diversas funcionalidades.

El SGCD gestiona datos sobre los clientes del centro, los entrenadores que imparten las clases deportivas, de dichas clases y de las pistas de entrenamiento y su uso está destinado al cargo de recepcionista del centro.

El diseño de este Sistema de Información nace de la necesidad de informatizar y procesar los datos en un centro deportivo debido a la gran cantidad de información que se puede llegar a crear para la gestión de este.

De cada cliente se almacenará su nombre y apellidos, su DNI, su teléfono, su correo electrónico, dirección y el estado de su suscripción. Para dar de alta a un nuevo cliente, el usuario introducirá estos datos en el sistema, confirmando la inserción o dando un error si así fuera. El DNI, el teléfono y el correo electrónico son únicos de cada cliente. Para dar de baja a un contacto, el usuario deberá introducir el DNI del cliente, confirmando el borrado o dando un error en caso de no encontrarse en el sistema. Para dejar constancia de la suscripción de un cliente, el usuario confirmará el pago manualmente, y el sistema se encargará de desactivarla pasado el plazo. Para apuntar un cliente a una clase, el usuario introducirá el ID de la clase e introducirá el DNI del cliente en la lista de alumnos de la clase. Para mostrar el listado completo de alumnos, el usuario enviará la petición al sistema y este le mostrará la lista.

De cada entrenador se almacenará su nombre y apellidos, su DNI, su teléfono, su correo electrónico, dirección y su especialidad. Para dar de alta a un nuevo entrenador, el usuario introducirá estos datos en el sistema, confirmando la inserción o dando un error si así fuera. El DNI, el teléfono y el correo electrónico son únicos de cada entrenador. Para dar de baja a un contacto, el usuario deberá introducir el DNI del entrenador, confirmando el borrado o dando un error en caso de no encontrarse en el sistema. Para dejar constancia del pago a un entrenador, el usuario confirmará el pago manualmente, y el sistema se encargará de desactivarlo pasado el plazo. Para mostrar el horario del entrenador, el usuario introducirá el día y el sistema mostrará el horario en forma de tabla. Para mostrar la lista de entrenadores, el usuario enviará la petición y el sistema mostrará el listado.

De cada pista se almacenará un identificador (IDpista), un aforo, las horas que está ocupada,el tipo de reserva y quien la ha reservado. Para reservar una pista, el usuario ingresará el IDpista, el tipo de reserva y el ID de quien reserva, el sistema confirmará dicha reserva o dará error. Para mostrar la lista de pistas, el usuario enviará la petición y el sistema mostrará el listado. Para mostrar el horario de reservas de la pista, el usuario introducirá el día y el sistema mostrará el horario en forma de tabla. Para cancelar una reserva, el usuario introducirá el IDpista de la pista y el sistema confirmará el trámite o dará error. Para mostrar la información de una reserva, el usuario introducirá el IDpista y la hora y el sistema mostrará todas las horas que está reservada y por quien está reservada.

De cada clase se almacenará un identificador (IDclase), número de participantes de la misma y la hora a la que se imparte. Para asignar un entrenador, el usuario introducirá el DNI del entrenador y el sistema confirmará el proceso si el entrenador tiene ese tramo horario libre. Para apuntar a un alumno, se introducirá el DNI del cliente y el sistema confirmará la inserción si aún quedan plazas para esa clase. Para reservar una pista, el usuario ingresará el IDpista y el sistema confirmará dicha reserva o dará error en el caso de que la pista esté ocupada en ese tramo horario. Para asignarle una hora, el usuario introducirá la hora y el sistema confirmará la asignación . Para asignarle una Temática, el usuario introducirá dicha temática y el sistema confirmará la inserción o dará error.

2. Subsistemas

En color verde se encuentran las correcciones.
En color magenta se encuentran las ampliaciones

1.-Clientes (Pérez Meneses, Francisco Alejandro)

RF1.1: Registro de clientes

Entrada:
Agente externo: usuario. Acción: solicitar creación de perfil de usuario.
Requisito de datos de entrada RDE1.1
BD: Requisitos de datos de escritura RDW1.1
Salida:
Agente externo: usuario. Acción: confirmación resultado.
Requisito de datos de salida: ninguno.

RDE1.1: Datos de entrada de alta del usuario
Nombre: Cadena de caracteres (30)
Apellidos: Cadena de caracteres (20)
DNI: Cadena de caracteres (9)
Dirección: Cadena de caracteres (40)
Número de teléfono: Cadena de caracteres (9)
Tipo de suscripción: Cadena de caracteres (9)
Solo podrá tomar los valores:
1. Normal (sólo podrá apuntarse a 5 clases al mes)
2. Medio (sólo podrá apuntarse a 15 clases al mes)
3. Premium (podrá apuntarse a 25 clases al mes)

RDW1.1: Datos almacenados del usuario
Mismos datos que RDE1.1

RF1.2: Dar de baja un cliente

Entrada:
Agente externo: cliente. Acción: solicitar borrado de perfil de cliente
Requisito de datos de entrada RDE1.2.
BD: Requisitos de datos de escritura RDW1.2
Salida:
Agente externo: cliente. Acción: confirmación resultado.
Requisito de datos de salida: ninguno.

RDE1.2: Datos de entrada de baja del cliente
DNI: Cadena de caracteres (9)

RDW1.2: Datos almacenados del usuario
Mismos datos que RDE1.2

RF1.3: Modificar datos de un cliente

Entrada:

Agente externo: cliente. Acción: solicitar modificar algún dato de su perfil
Requisito de datos de entrada RDE1.3.
BD: Requisitos de datos de escritura RDW1.3
Salida:
Agente externo: administrador. Acción: confirmación resultado.
Requisito de datos de salida: ninguno.

RDE1.3: Datos de entrada de modificación del cliente (*)
Nombre: Cadena de caracteres (30)
Apellidos: Cadena de caracteres (20)
DNI: Cadena de caracteres (9)
Dirección: Cadena de caracteres (40)
Número de teléfono: Cadena de caracteres (9)

RDW1.3: Datos almacenados del usuario
Mismos datos que RDE1.3

(*)En caso de querer modificar un solo campo, el cliente podrá elegir cuál.

RF1.4: Gestionar la suscripción del cliente

Entrada:

Agente externo: cliente. Acción: solicitar modificación de su suscripción
Requisito de datos de entrada RDE1.4.
BD:
Requisitos de datos de escritura RDW1.4
Requisitos de datos de lectura RDR1.4
Salida:
Agente externo: administrador. Acción: confirmación resultado.
Requisito de datos de salida: ninguno.

RDE1.4: Datos de entrada de modificación del cliente
DNI: Cadena de caracteres (9)
Tipo de suscripción: Cadena de caracteres (9)

RDW1.4: Datos almacenados del usuario
Mismos datos que RDE1.4

RDR1.4: Tipo de suscripción del cliente

RF1.5: Apuntar un cliente a una clase

Entrada:

Agente externo: cliente. Acción: solicitar reserva de clase

Requisito de datos de entrada RDE1.5.

BD:

Requisitos de datos de escritura RDW1.5

Salida:

Agente externo: administrador. Acción: confirmación resultado.

Requisito de datos de salida: ninguno.

RDE1.5: Datos de entrada de modificación del cliente

DNI: Cadena de caracteres (9)

ID de la clase: Cadena de caracteres (9)

RDW1.5: Datos almacenados del usuario

Mismos datos que RDE1.5

2.-Entrenadores (Hernández Aranguren, Pedro)

RF2.1: Creación entrenador

Entrada:

Agente externo: usuario. Acción: solicitar la creación de un nuevo entrenador.
Requisito de datos de entrada RDE2.1
BD: Requisitos de datos de escritura RDW2.1
Salida: Agente externo: usuario. Acción: confirmación resultado.
Requisito de datos de salida: ninguno.

RDE2.1: Datos de entrada de creación de un entrenador
DNI: Cadena de caracteres (9)
Nombre: Cadena de caracteres (20)
Apellidos: Cadena de caracteres (40)
Correo electrónico: Cadena de caracteres (60)
Dirección: Cadena de caracteres (60)
Número de teléfono: Cadena de caracteres (9)
Especialidad: Menú de opciones (natación, atletismo, gimnasio, actividades)

RDW2.1: Datos almacenados del usuario
Mismos datos que RDE2.1

RF2.2: Borrar un entrenador

Entrada: Agente externo: usuario. Acción: solicitar borrado de entrenador.
Requisito de datos de entrada RDE2.2.
BD: Requisitos de datos de escritura RDW2.2
Salida: Agente externo: usuario. Acción: confirmación resultado.
Requisito de datos de salida: ninguno.

RDE2.2: Datos de entrada de baja del entrenador
DNI: Cadena de caracteres (9)

RDW2.2: Datos almacenados del usuario
Mismos datos que RDW2.1

RF2.3: Calcular el pago de un entrenador

Entrada: Agente externo: usuario. Acción: solicitar el cálculo de un entrenador.
Requisito de datos de entrada RDE2.3.
BD: Requisitos de datos de lectura RDR2.3. y de escritura RDW2.3.
Salida:
Agente externo: usuario. Acción: confirmación resultado. Requisito de datos de salida: ninguno.

RDE2.3: Datos de entrada de calculo del salario del entrenador
DNI: Cadena de caracteres (9)

RDR2.3: Consulta del horario del entrenador

RDW2.3: El cálculo del salario efectuado del entrenador.
salario: entero

RF2.4: Devolver el horario de un entrenador

Entrada: Agente externo: usuario. Acción: solicitar el horario de un entrenador.
Requisito de datos de entrada RDE2.4.
BD: Requisitos de datos de lectura RDR2.4.
Salida: Agente externo: usuario. Acción: confirmación resultado.
Requisito de datos de salida: RDS2.4.

RDE2.4: Datos de entrada del horario de un entrenador
DNI: Cadena de caracteres (9)
Día de la semana: Menú (lunes, martes, miércoles, jueves, viernes)

RDR2.4: Las horas de clase que da un entrenador.
DNI del entrenador: Cadena de caracteres (9)
ID de la clase : Cadena de caracteres (20)
Nombre de entrenador: Cadena de caracteres (20)
Horario de la clase : date

RDS2.4: El horario del entrenador en el día seleccionado.
DNI del entrenador: Cadena de caracteres (9)
Nombre de la clase: Cadena de caracteres (20)
Nombre de entrenador: Cadena de caracteres (20)
Horario de la clase : date

RF2.5: Devolver un listado de entrenadores

Entrada:
Agente externo: usuario. Acción: solicitar un listado de entrenadores. Requisito de datos de entrada: ninguno
BD: Requisitos de datos de lectura RDR2.5.
Salida:
Agente externo: usuario. Acción: confirmación resultado. Requisito de datos de salida: RDS2.5.

RDR2.5: Los entrenadores almacenados.
DNI: Cadena de caracteres (9)
Nombre: Cadena de caracteres (20)
Apellidos: Cadena de caracteres (40)
Especialidad: Menú de opciones (natación, atletismo, gimnasio, actividades)

RDS2.5: Un listado de registros, cada uno con los datos de RDR2.5.

3.-Clases (Heredia Muñoz, José)

RF3.1: Asignación de una instalación para la clase.

Entrada: Agente externo: usuario.
RDE3.1.
Acción: asignar instalación a una clase.
BD: RDW3.1 ,RDR3.1
Salida:
Requisitos de datos de salida: ninguno.

RDE3.1: Datos de la instalación.
ID instalación: Cadena de caracteres (20).
ID clase: Cadena caracteres (20).

RDW3.1: Asignamos instalación a una clase
Tipo de dato: ID instalación: Cadena caracteres(20)
ID clase: Cadena caracteres(20)

RDR3.1: Consulta disponibilidad de la instalación
Consulta aforo
Consulta horario a la clase
Consulta disponibilidad Clase-Instalación
Tipo de dato: ID instalación: Cadena caracteres(20)
Aforo: entero
Horario: date
ID clase: Cadena caracteres(20)

RF3.2: Asignación de un entrenador para la clase.

Entrada: Agente externo: usuario.
RDE3.2
Acción: asignar entrenador a una clase.
BD: RDW2.3, RDR3.2
Salida:
Requisitos de datos de salida: ninguno.
RDE3.2: Datos entrenador
DNI: Cadena caracteres (9).
ID clase: Cadena caracteres (20).
RDW3.2: Asignamos el entrenador a la clase.
Tipo de dato: ID clase: Cadena caracteres (20).
DNI Entrenador: Cadena caracteres (20)

RDR3.2: Consulta disponibilidad del entrenador en la tabla Clase-Entrenador.
Consulta horario clase.
ID clase: Cadena caracteres (20).
DNI Entrenador: Cadena caracteres (20).

RF3.3: **Crear una clase.**

Entrada: Agente externo: usuario

Acción: Crear la clase

BD: RDW3.3

Salida:

Requisitos de datos de salida: ninguno.

RDW3.3: Se le asigna una temática a la clase.
Se le asigna un horario a la clase

Tipo de dato: Temática: Cadena de caracteres(20).
Horario: Date(20).

RF3.4: **Listar clase/s**

Entrada: Agente externo: usuario RDE3.4.

Acción: Muestra una o varias clases con todos sus atributos

BD: RDR3.4.

Salida:

Requisitos de datos de salida: ninguno.

RDE3.4: - Si no se proporciona nada, muestra todas las clases.
-ID de la clase. Muestra

RDR3.4: Muestra Temática-Horario-Instalación-Entrenador.

ID clase: Cadena caracteres (20).
Temática: Cadena de caracteres(20).
Horario: Date(20).
ID instalación: Cadena caracteres(20)

RF3.5: **Borrar clase.**

Entrada: Agente externo: usuario.

Acción: Se eliminan todos los campos de una clase de todas las tablas RDE3.5.

BD: RDW3.5.

Salida:

Requisitos de datos de salida: ninguno.

RDE3.5: -ID Clase

RDW3.5: Se borran los datos de clase.

Cambio sustancial en el subsistema de clases: He cambiado algunos requisitos funcionales con la intención de hacer un diseño más coherente y robusto.
He añadifo el RF- Crear clase, donde directamente se le asigna una temática y un horario que es lo que quiero que defina a una clase, pues en función de estos 2 atributos se asignará los entreadores y las instalaciones.
Osea, solo puede haber una clase de X temática a cierta hora
He quitado el RF: Inscribir cliente porque considero que este no era el subsistema donde correspondia esa funcionalidad, eso se ha de hacer desde el subsistema de clientes.
He añadido los RF: Listar clases-Borrar calse, para poder tener mejor manejo sobre el número de clases que hay, de cara a informar al cliente-gerente y poder borrar alguna si no se encuentra entrenador o instalación que asignarle

4.-Instalaciones (Salazar Cano, Santiago)

RF4.1: Reserva de una pista por un cliente.

Entrada: Agente externo: usuario.
RDE1.
Acción: asignar instalación a un cliente.
BD: requisito de datos de escritura RDW1.
Salida: Agente externo: usuario.
Requisitos de datos de salida: ninguno.

RDE1: Datos de la instalación, IDcliente
Nombre: Cadena de caracteres (20).
RDW1: Alteración del estado de la instalación, pasa a estar ocupada en ese tramo horario.

RF4.2: Mostrar lista de pistas existentes

Entrada: Agente externo: usuario.
RDE2.
Acción: Mostrar una lista de pistas existentes en el centro
BD: requisito de datos de lectura RDE2.
Salida: Agente externo: usuario.
Requisitos de datos de salida: ninguno.

RDE2: Datos de cada pista:
Nombre: Cadena de caracteres (20).
IDpista: Cadena de caracteres (5)

RF4.3: El subsistema mostrará el horario de una pista

Entrada: Agente externo: usuario RDE3
Acción: Muestra el horario de las pistas en un dia
BD: requisito de datos de lectura
Salida: Agente externo: usuario.
Requisitos de datos de salida: permiso de lectura

RDE3: Datos de la pista.
IDpista:Cadena de caracteres(5).
Fecha: Date

RDS3: Por cada reserva con la pista #Pista
#Cliente: int
Hora: Date

RF4.4: El subsistema mostrará la información de una reserva en específico

Entrada: Agente externo: usuario RDE4
Acción: mostrar datos de la reserva.
BD: requisito de datos de lectura
Salida: Agente externo: usuario.
Requisitos de datos de salida: lectura

RDE4: Datos del cliente.
IDpista: Cadena Caracteres (5)
Hora: date

RDS4: Datos del cliente.
Nombre: Cadena de caracteres (20).
Apellidos: Cadena caracteres (40).
DNI: Cadena caracteres (9).

Datos de la pista.
Horas reservadas: Vector de enteros (16)

RF4.5: Cancelación de reserva

Entrada: Agente externo: usuario RDE5
Acción: Controlar el aforo de cada clase modificando cuanta gente hay asignada.
BD: requisito de datos de escritura RDW5.1
Salida: Agente externo: usuario.
Requisitos de datos de salida: ninguno.

RDE5: Datos del cliente.
IDpista: Cadena Caracteres (20)
Hora: Date

RDW5.1: Eliminación de la reserva en las horas de la pista.

5.-Restricciones semánticas

RS1.1: El cliente no debe existir en el sistema.
RF: RF1.1
RD(s): RDE1.1

Descripción: “Es necesario que no exista ningún otro cliente con los mismos datos, es decir, DNI, para que no haya problemas de identidad”.

RS1.2: El cliente debe existir en el sistema.
RF: RF1.2, RF1.3, RF1.4, RF1.5
RD(s): RDE1.2, RDE1.3, RDE1.4, RDE1.5

Descripción: “Es necesario que el cliente exista en la base de datos para poder modificar o borrar sus datos, además de las tareas relacionadas con apuntarse a una clase”.

RS1.3: El cliente debe existir en el sistema.
RF: RF1.3
RD(s): RDE1.3, RDE1.4, RDE1.5

Descripción: “Es necesario que el cliente exista en la base de datos para poder modificar o borrar sus datos, además de las tareas relacionadas con apuntarse a una clase”.

RS1.4: El cliente debe existir en el sistema.
RF: RF1.4
RD(s): RDE1.4

Descripción: “Es necesario que el cliente exista en la base de datos para poder modificar o borrar sus datos, además de las tareas relacionadas con apuntarse a una clase”.

RS1.5: El cliente debe existir en el sistema.
RF: RF1.5
RD(s): RDE1.5

Descripción: “Es necesario que el cliente exista en la base de datos para poder modificar o borrar sus datos, además de las tareas relacionadas con apuntarse a una clase”.

RS1.6: La clase debe existir en el sistema.
RF: RF1.5
RD(s): RDE1.5

Descripción: “La clase ha de estar en la base de datos del sistema para poder que un cliente pueda ser apuntado a esta”.

RS1.7: La clase no debe haber alcanzado el aforo máximo permitido.
RF: RF1.5
RD(s): RDE1.5

Descripción: “Si la clase está completa, no se permitirá la reserva o entrada de nuevos clientes a esta, debido a las normativas de las instalaciones”.

RS2.1: No pueden existir dos entrenadores con el mismo DNI
RF: RF2.1
RD(s) RDW2.1

Descripción: “Si ya había un entrenador con el mismo DNI, no se inserta el nuevo entrenador y devuelve un error.”

RS2.2: No pueden existir dos entrenadores con el mismo teléfono
RF: RF2.1
RD(s) RDW2.1

Descripción: “Si ya había un entrenador con el mismo teléfono, no se inserta el nuevo entrenador y devuelve un error.”

RS2.3: No pueden existir dos entrenadores con el mismo correo
RF: RF2.1
RD(s) RDW2.1

Descripción: “Si ya había un entrenador con el mismo correo, no se inserta el nuevo entrenador y devuelve un error.”

RS2.4: No se puede borrar un entrenador que no exista
RF: RF2.2
RD(s) RDE2.2

Descripción: “Si al ingresar el DNI se comprueba que no existe, devuelve un error.”

RS2.5: No se puede calcular el salario de un entrenador que no exista
RF: RF2.3
RD(s) RDE2.3

Descripción: “Si al ingresar el DNI se comprueba que no existe, devuelve un error.”

RS2.6: No se puede devolver el horario de un entrenador que no exista
RF: RF2.4
RD(s) RDE2.4

Descripción: “Si al ingresar el DNI se comprueba que no existe, devuelve un error.”

RS3.1: La instalación ha de estar libre, si no lo estuviese se devuelve un error.
RF:RF3.1

RS3.2: El entrenador ha de estar libre en ese tramo horario, si no lo estuviese, se devuelve un error.
RF:RF3.2

RS4.1: La instalación ha de estar libre, si no lo estuviese se devuelve un error.
RF: RF4.1
RD(s): RDE1

RS4.2: Si no existe la pista con el IDpista introducido, se devuelve un error.
RF: RF4.3
RD(s): RDE2

RS4.3: **Si se ha alcanzado el aforo máximo para esa clase, no se realiza la inscripción y se devuelve un error.**
RF: RF4.4
RD(s): RDE4

RS4.4: **Si no hay reserva a esa hora, el subsistema notifica el error.**
RF: RF4.5
RD(s): RDE5

3. Paso a Tablas

Tablas del subsistema de clientes.

Clientes(#DNI, Nombre, Apellidos, #correo, Dirección, #Teléfono, tipo_suscripcion)
Apuntado(#DNI, #ID_clase)

Tablas del subsistema entrenadores.

Entrenadores(#DNI, Nombre, Apellidos, #correo, Dirección, #Teléfono, Especialidad, salario)

Tablas del subsistema clases.

Clase(#IDclase,Temática,Horario)

Dependencias funcionales completas: {IDclase -> temática, IDclase -> horario}
Atributos no primos: temática y horario.
Esta relación ya está en 2FN.

Imparte(#IDclase,#DNI)

Se_da(#IDclase,#IDinstalación)

Fusión de tablas.

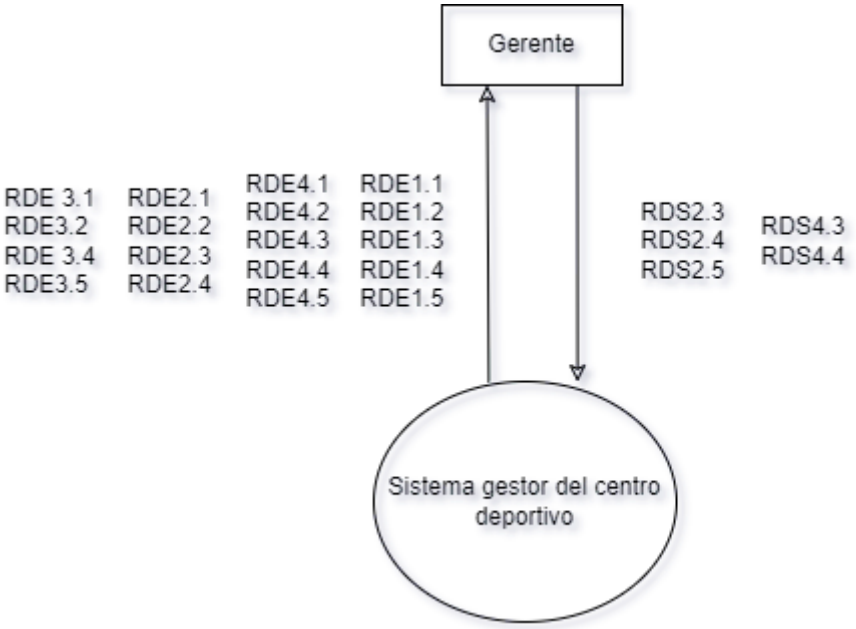
Clase(#IDclase,Temática,Horario,#DNI,#IDinstalación)
Esta relación está en 2FN ya que la CP #IDclase determina a los demás atributos, que son no primos, de forma completa.

Tablas del subsistema de instalaciones

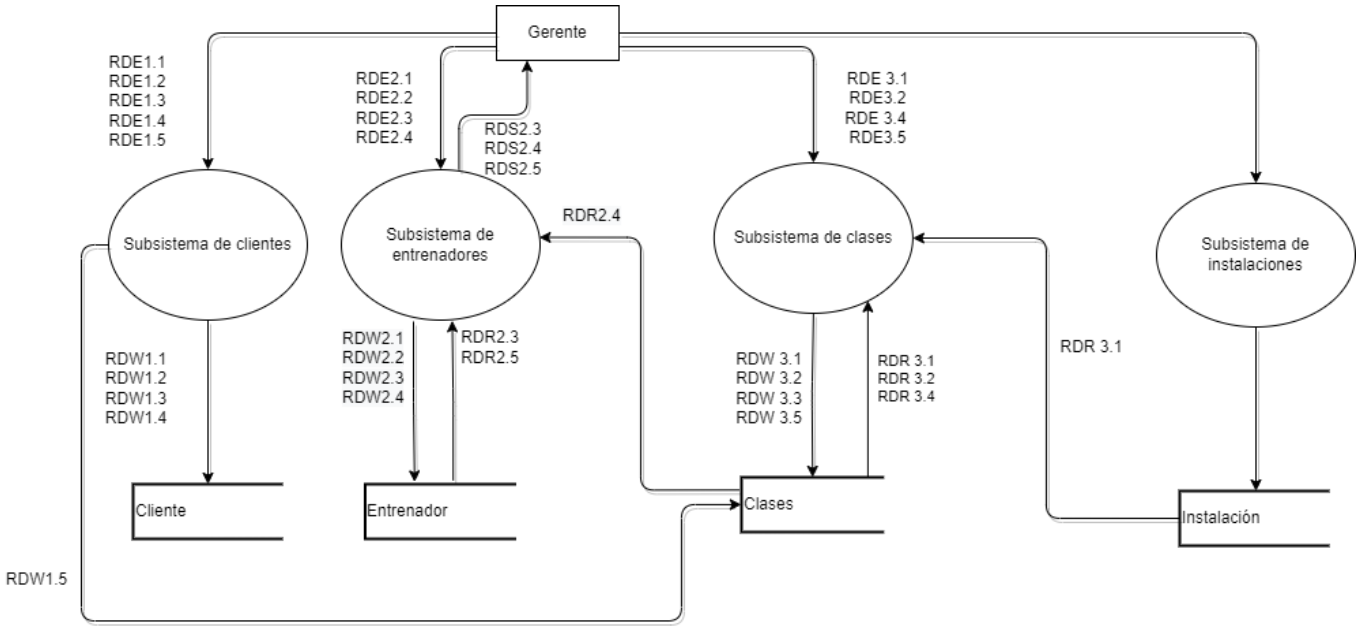
Instalación(#IDinstalación,Nombre)
Reserva(#IDinstalación,#DNI, Fecha)
Fusión de tablas -> Esta relación está en 2FN ya que la CP #IDinstalación determina a los demás atributos, que son no primos, de forma completa

4. Diagramas

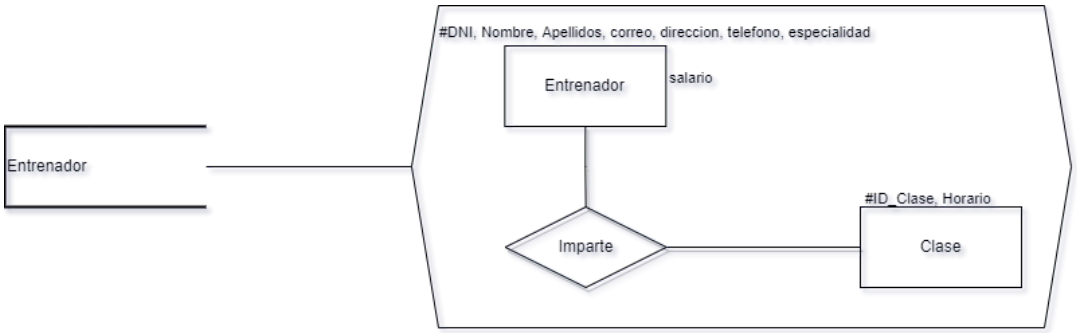
Caja negra

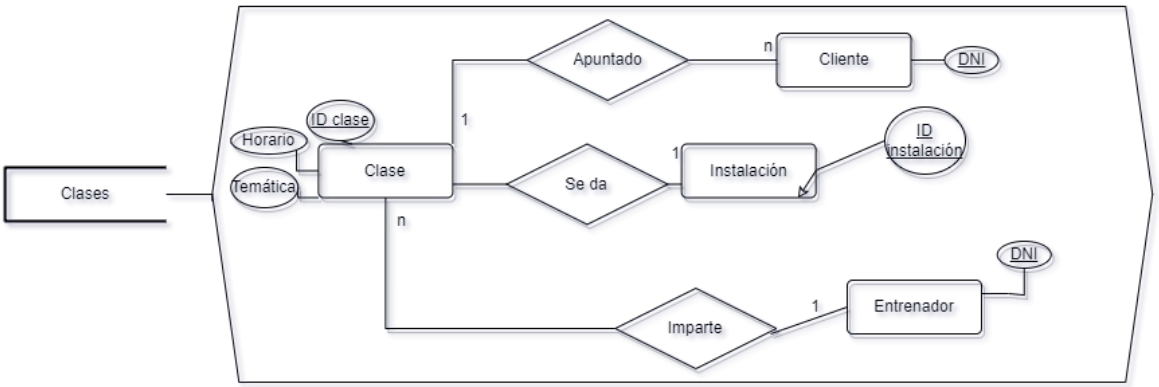


DFD0 o Armazón

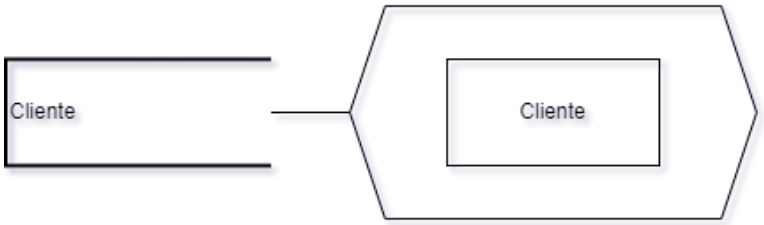


Esquema externo del DFD0



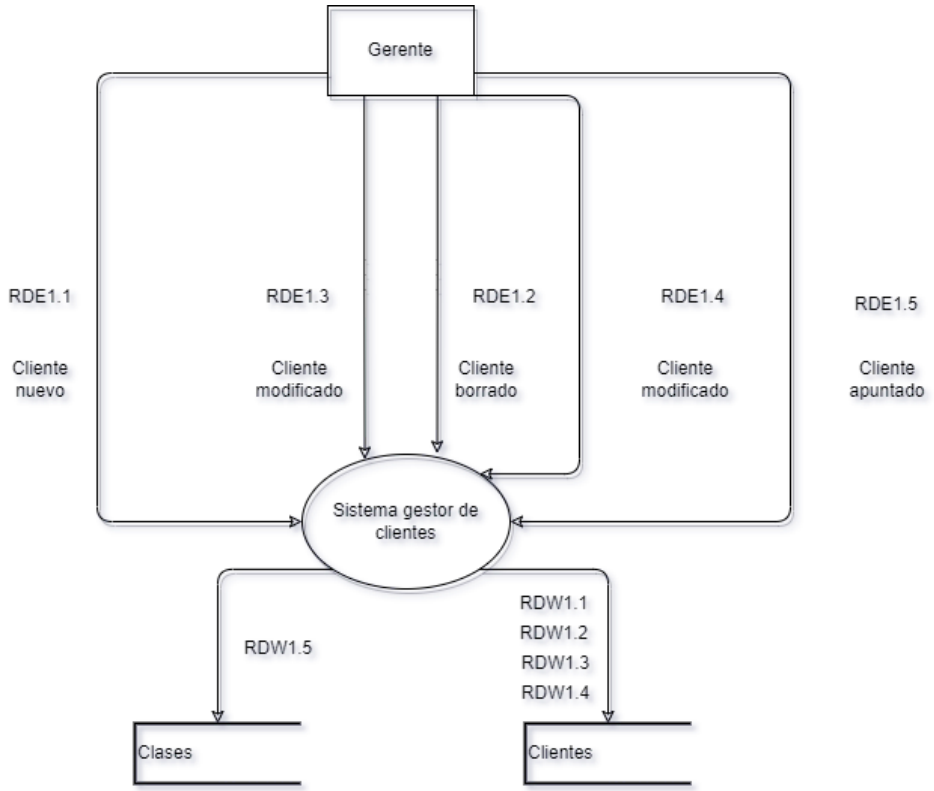


#DNI, Nombre, Apellidos, correo, dirección, teléfono, tipo_suscripción

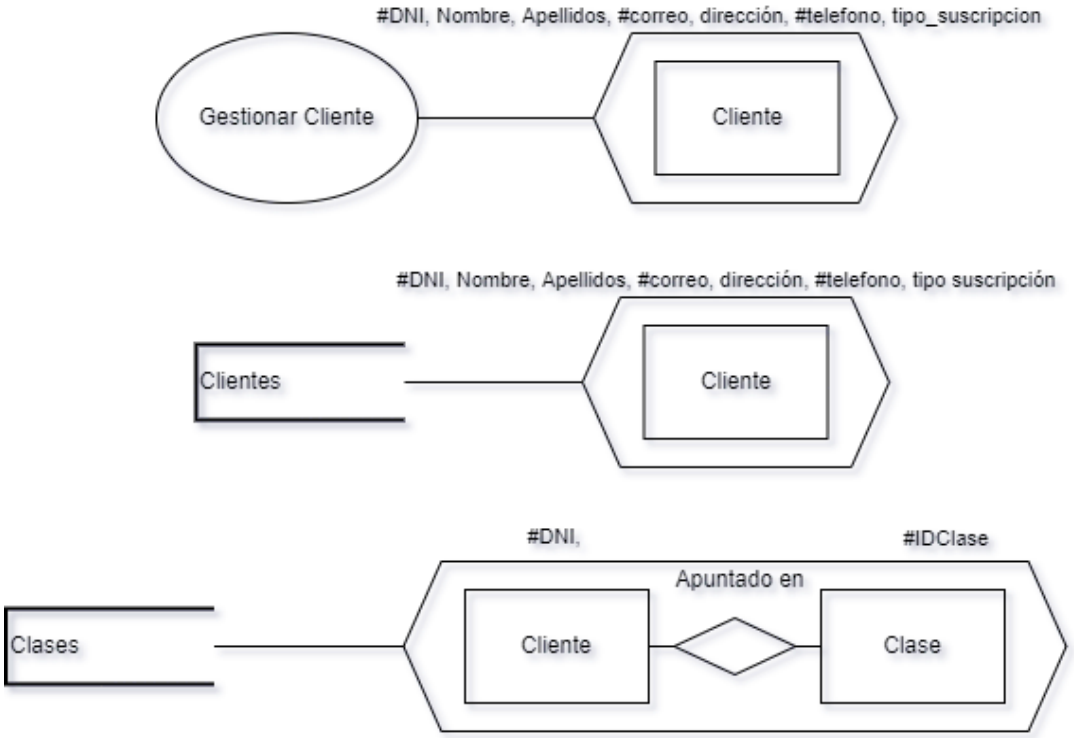


Clientes

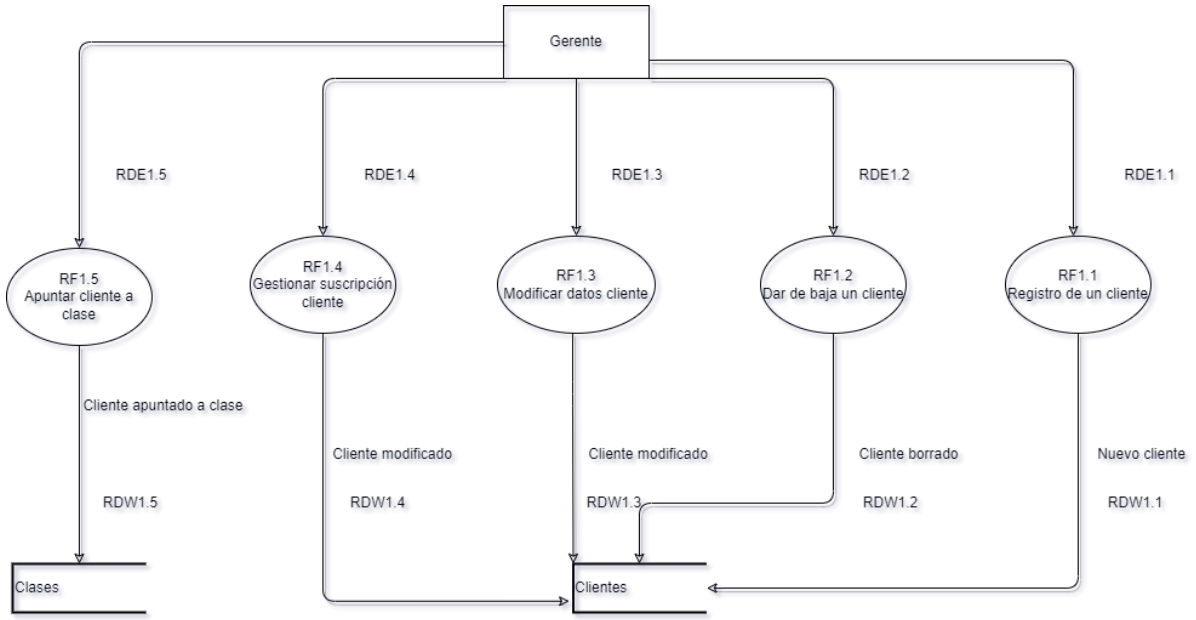
DFD0



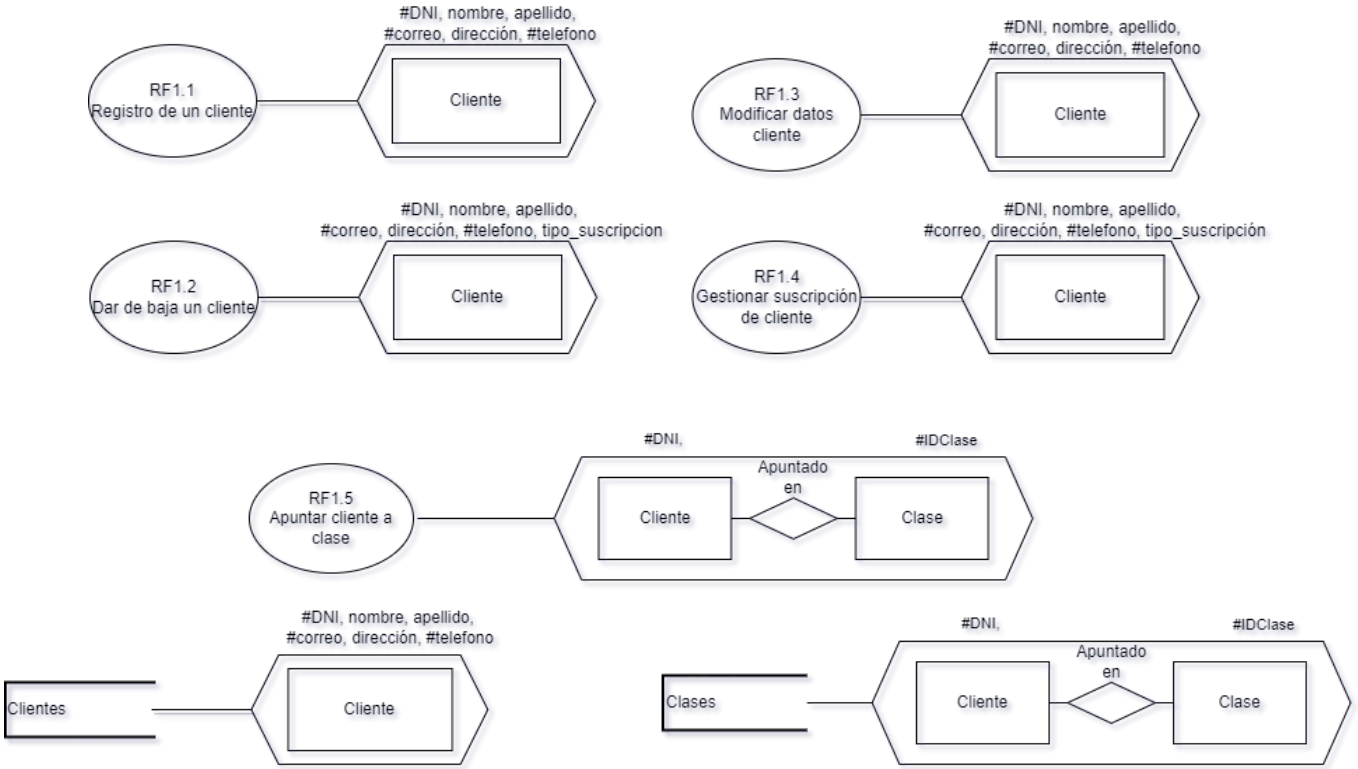
Esquema externo del DFD0



DFD1

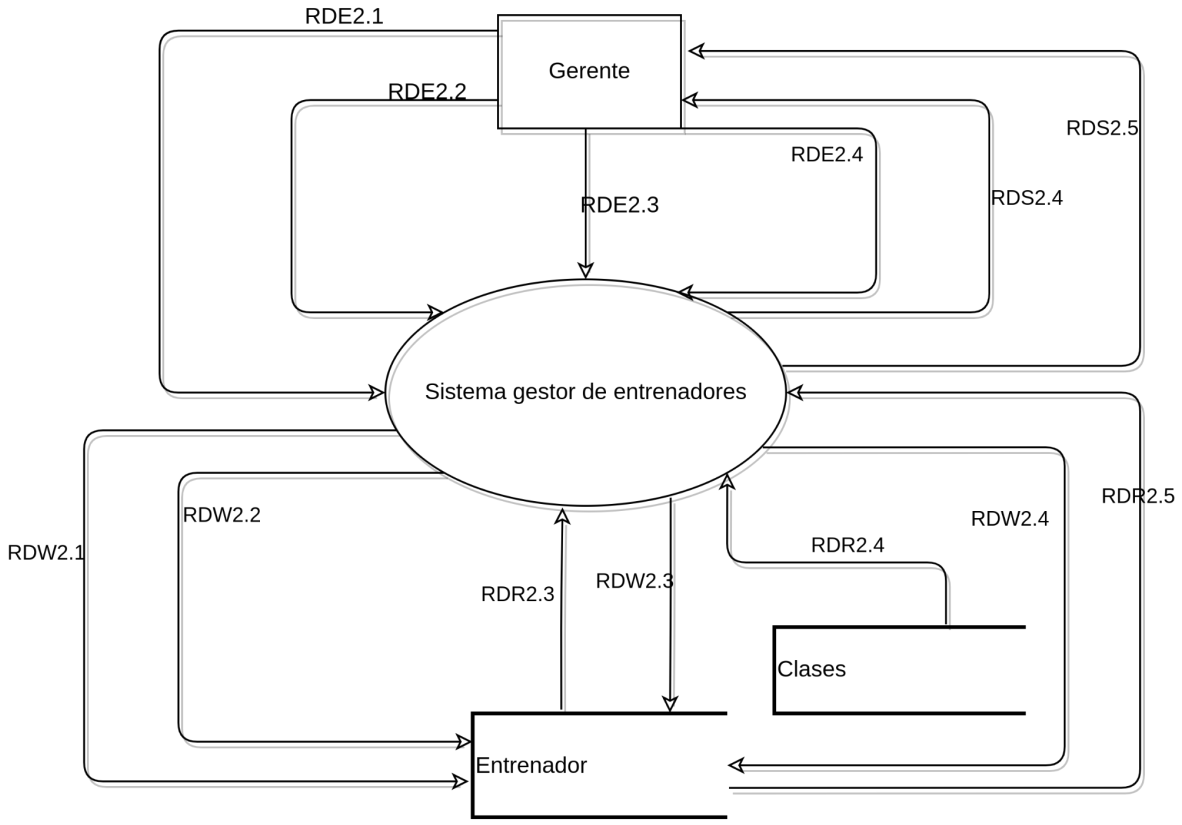


Esquema externo del DFD1

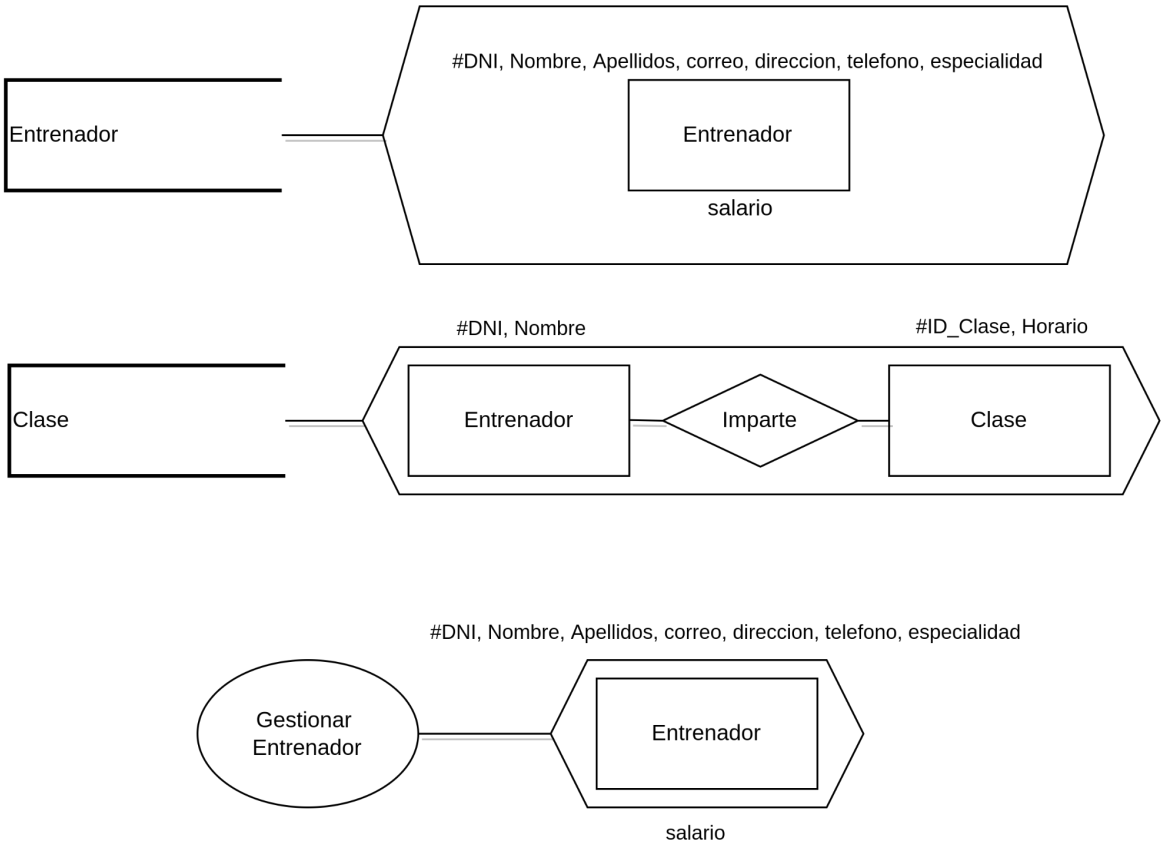


Entrenadores

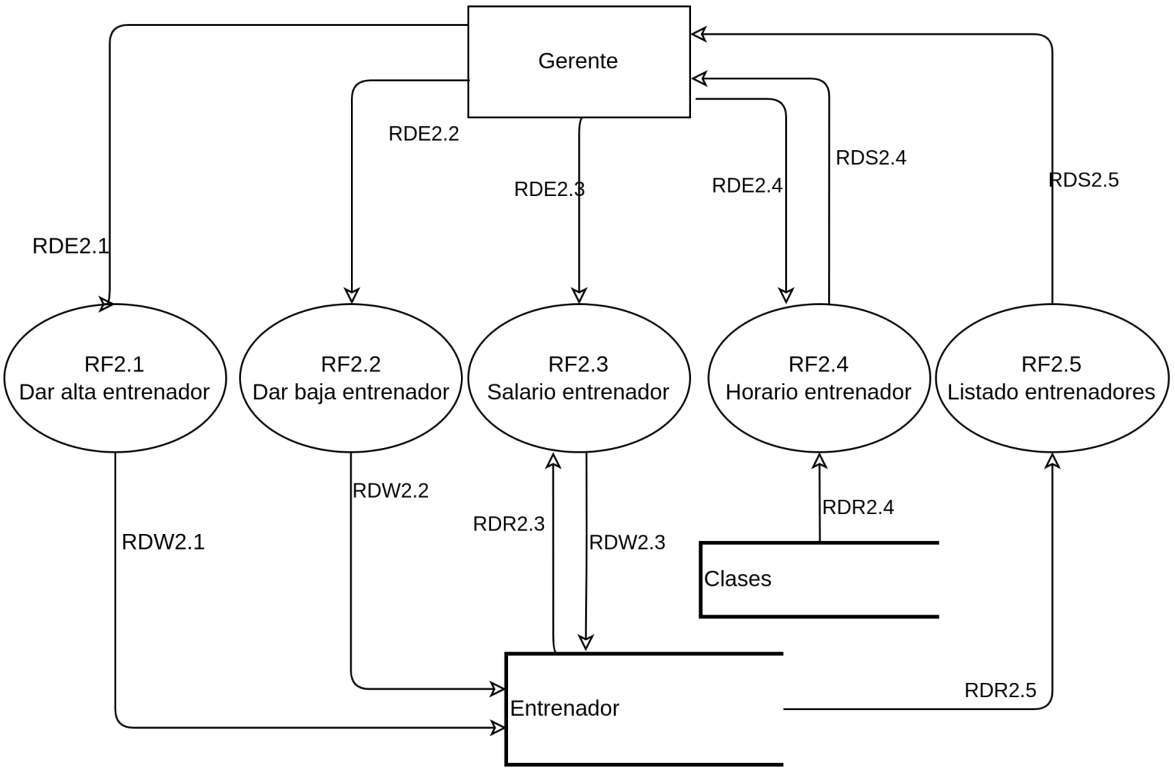
DFD0



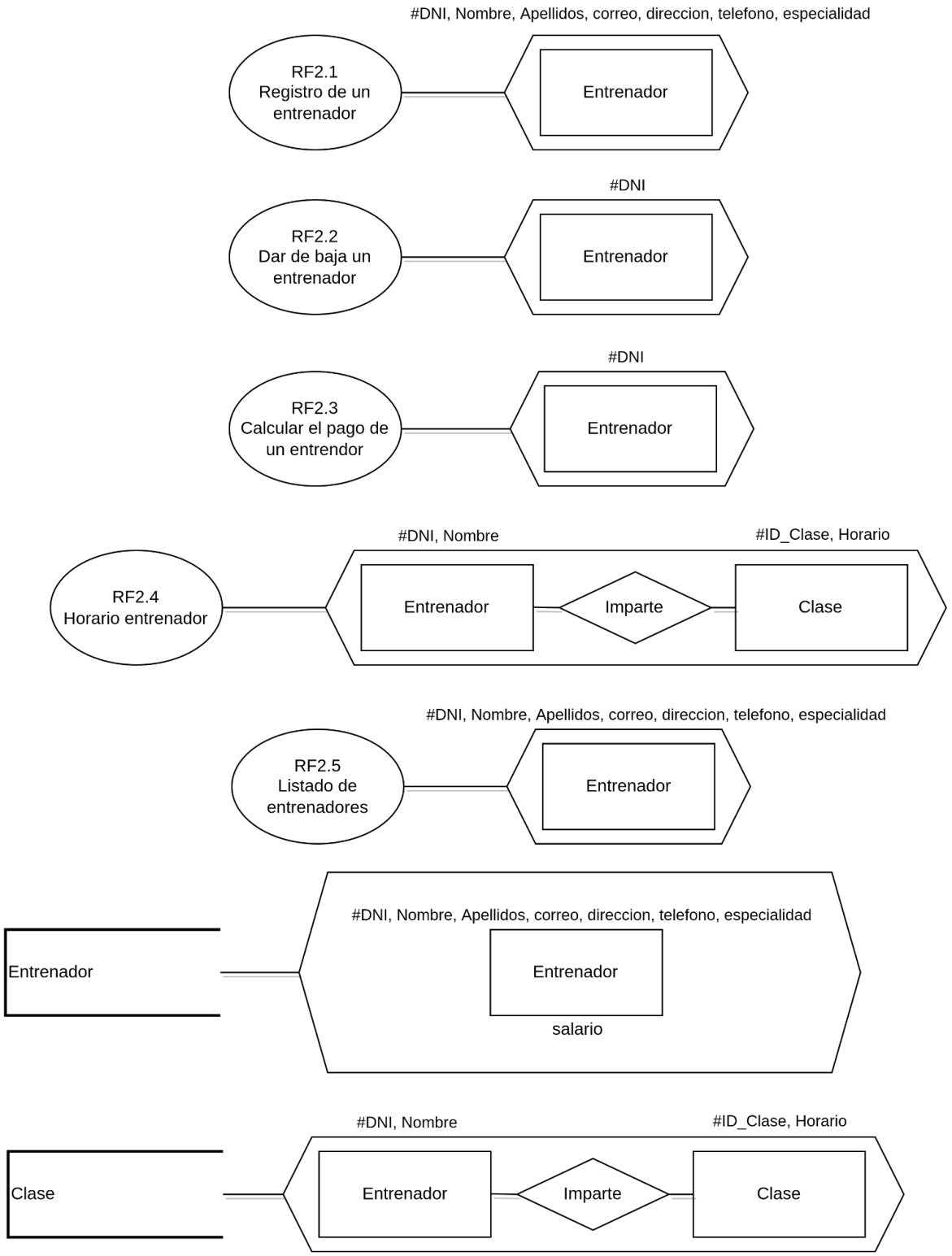
Esquema externo del DFD0



DFD1

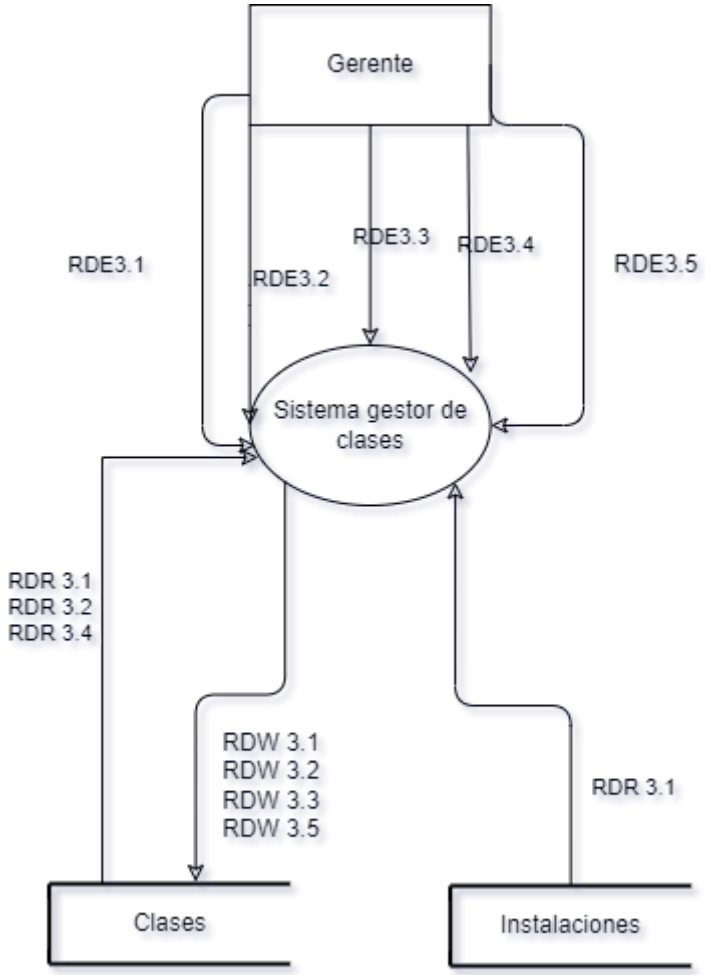

Gerente
RDE2.1
RDE2.2
RDE2.3
RDE2.4
RDS2.4
RDS2.5
RF2.1
Dar alta entrenador
RF2.2
Dar baja entrenador
RF2.3
Salario entrenador
RF2.4
Horario entrenador
RF2.5
Listado entrenadores
RDW2.1
RDW2.2
RDR2.3
RDW2.3
RDR2.4
Clases
Entrenador
RDR2.5

Esquema externo del DFD1

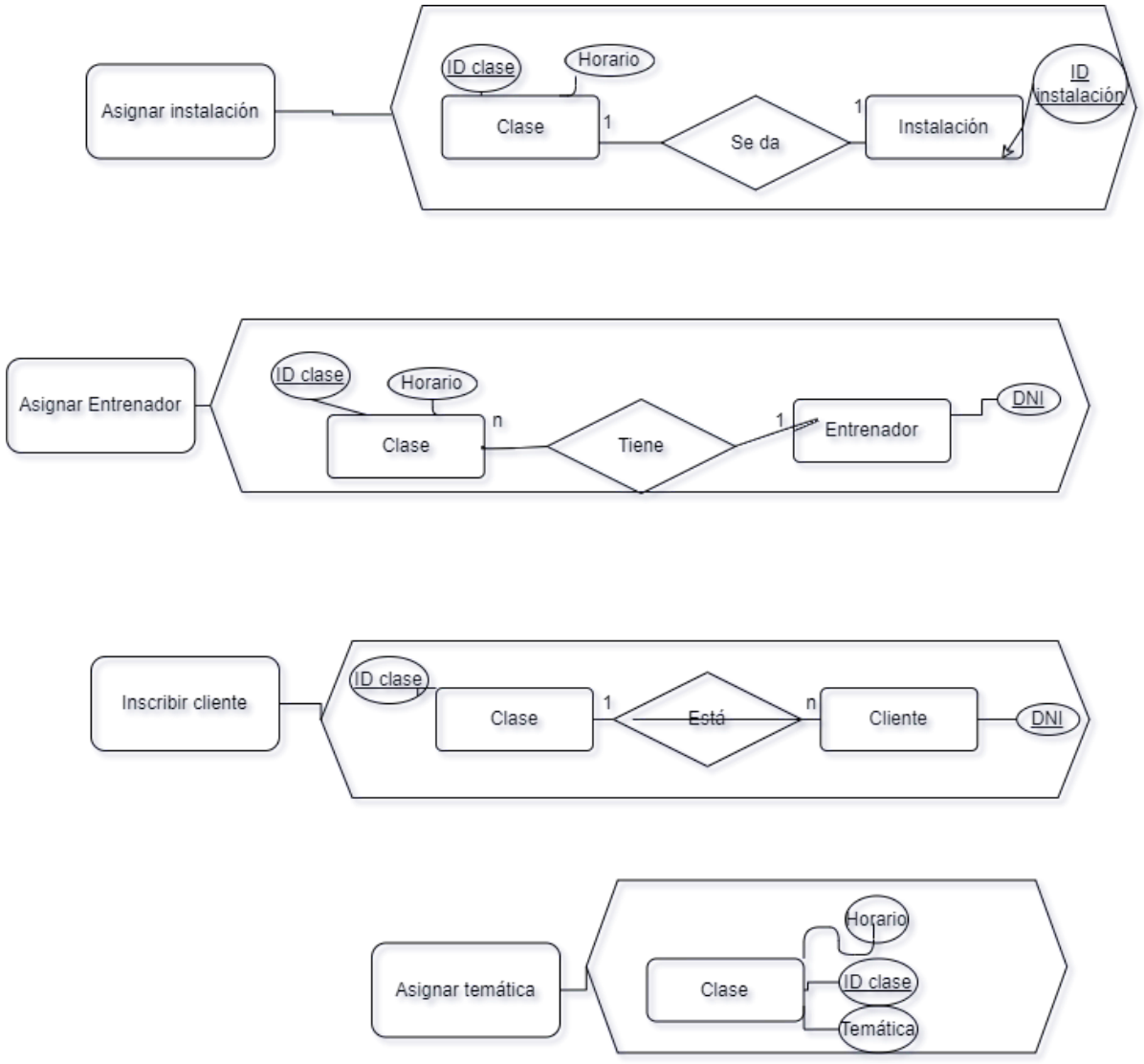


Clases

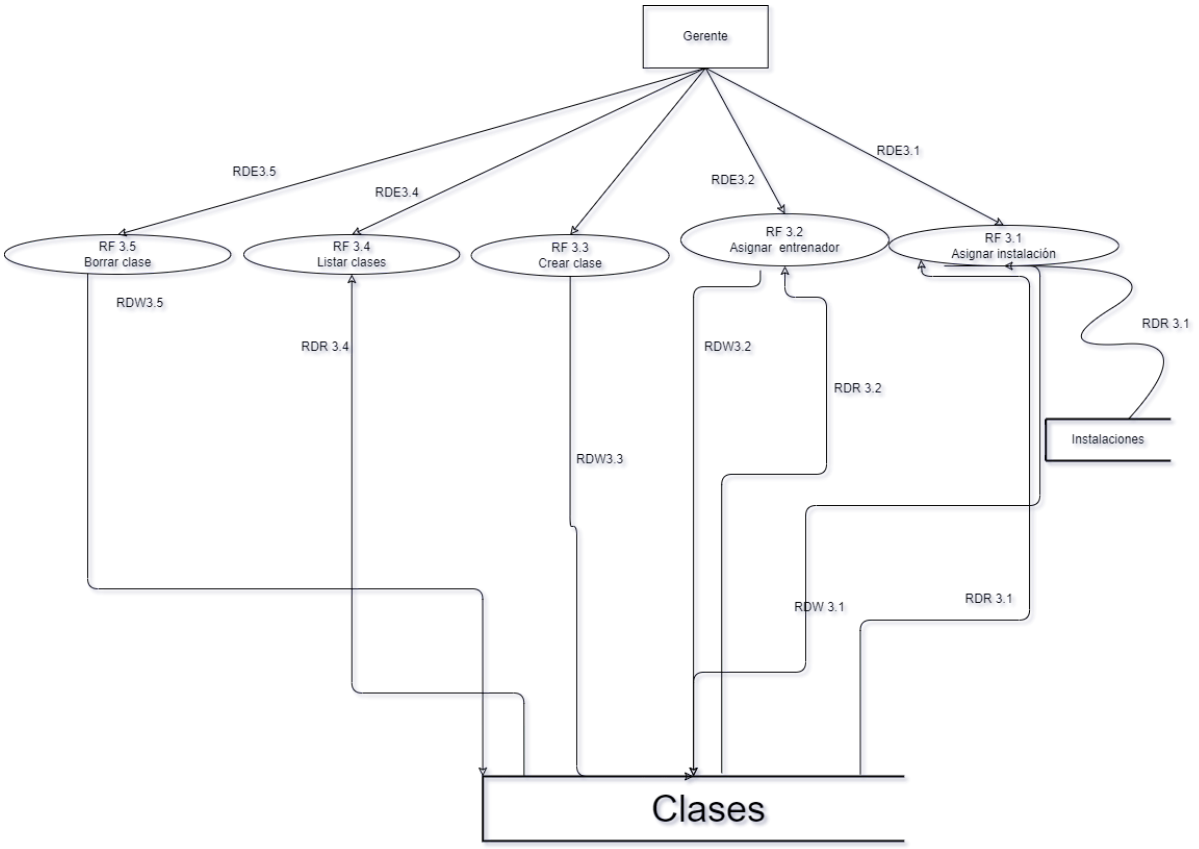
DFD0


Gerente
RDE3.1
RDE3.2
RDE3.3
RDE3.4
RDE3.5
Sistema gestor de clases
RDR 3.1
RDR 3.2
RDR 3.4
RDW 3.1
RDW 3.2
RDW 3.3
RDW 3.5
RDR 3.1
Clases
Instalaciones

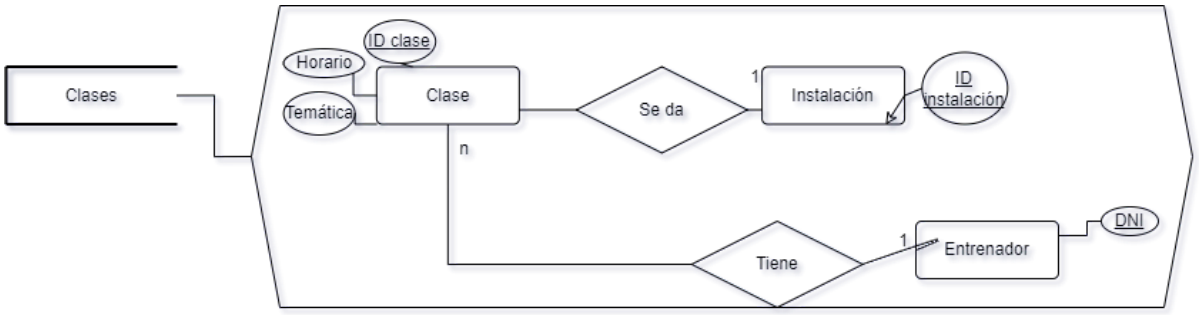
Esquema externo del DFD0



DFD1

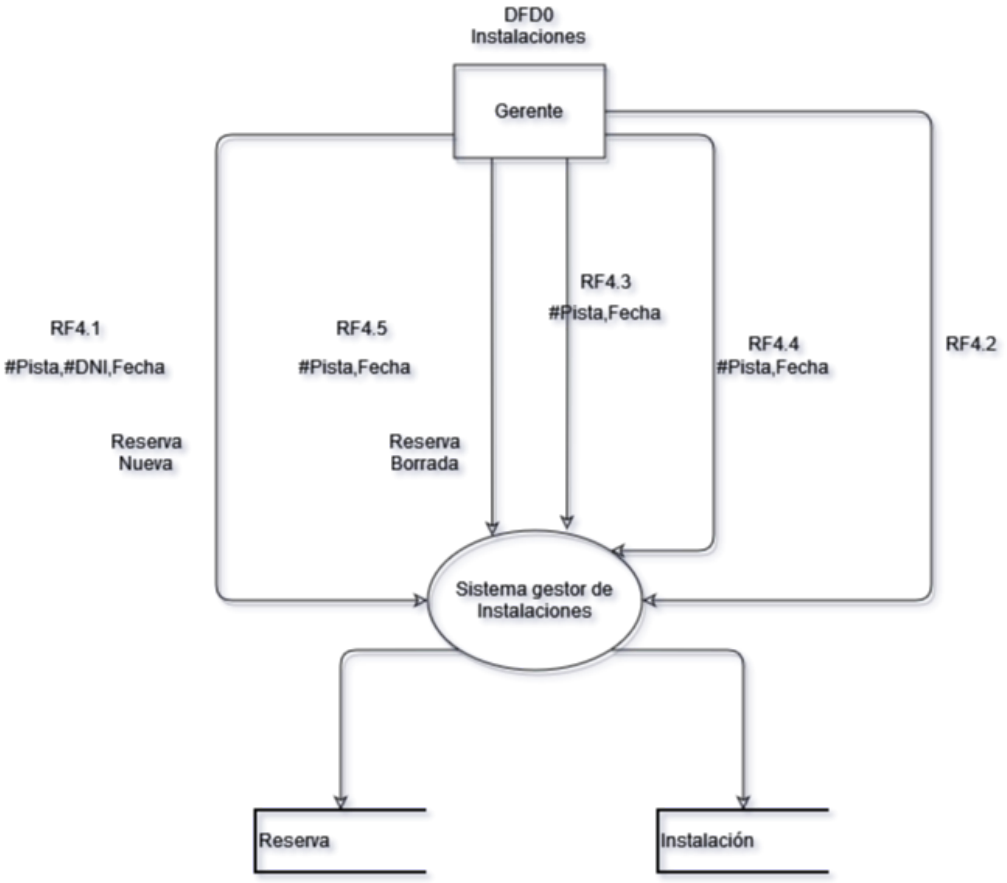


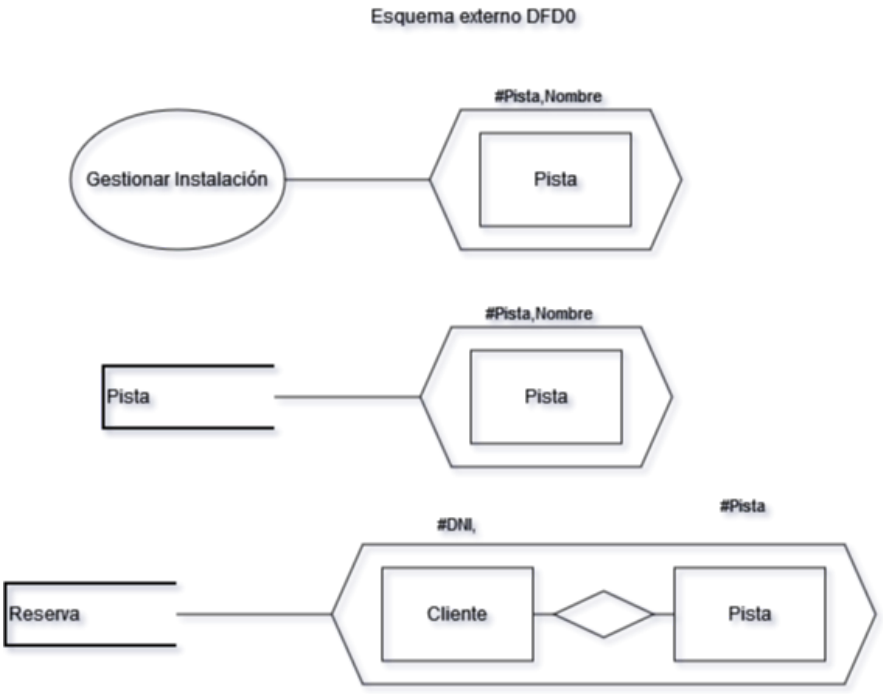
Esquema externo del DFD1



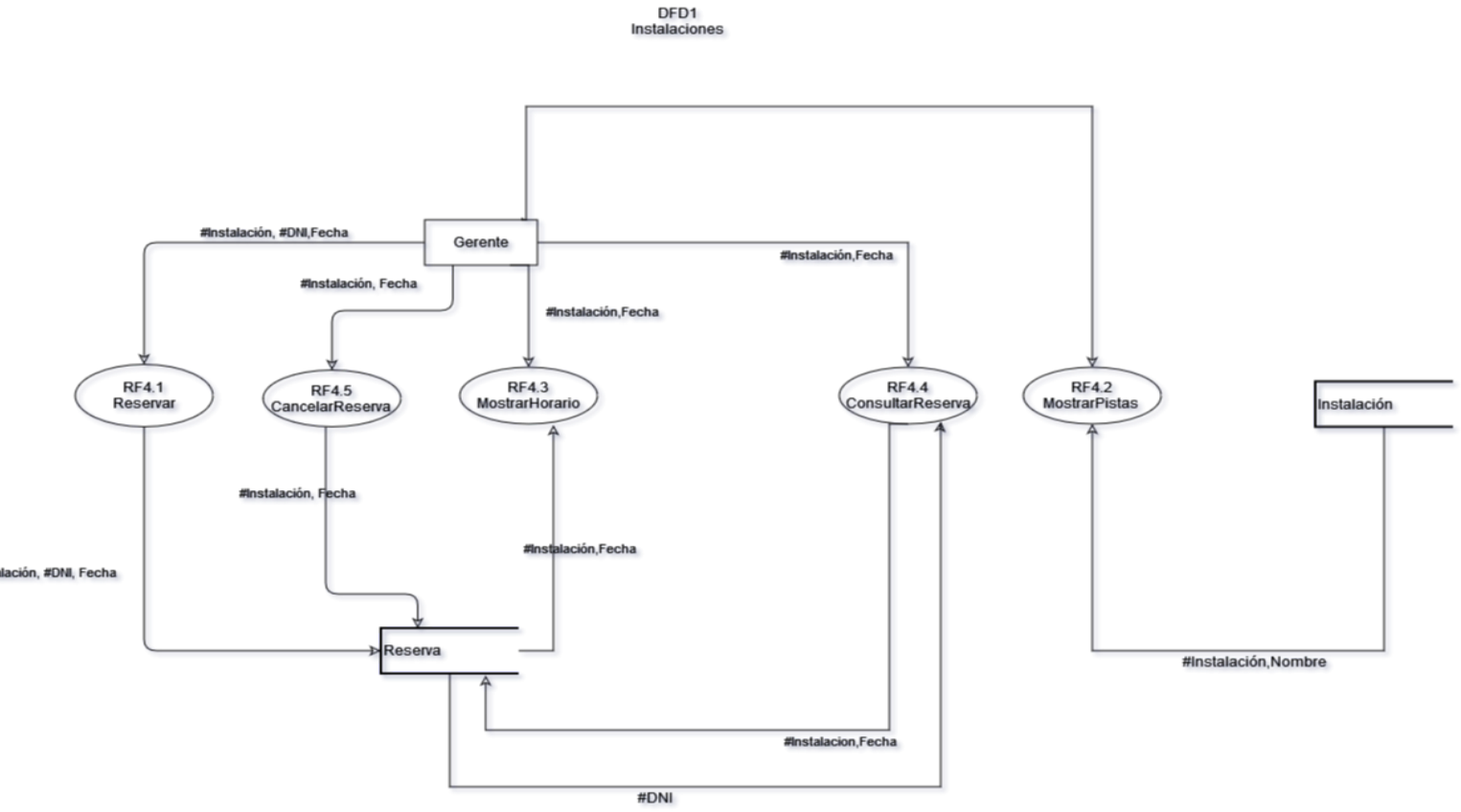
Instalaciones

DFD0





DFD1


DFD1
Instalaciones
Gerente
#Instalación, #DNI,Fecha
#Instalación, Fecha
#Instalación,Fecha
#Instalación,Fecha
RF4.1
Reservar
RF4.5
CancelarReserva
RF4.3
MostrarHorario
RF4.4
ConsultarReserva
RF4.2
MostrarPistas
Instalación
#Instalación, Fecha
#Instalación,Fecha
lación, #DNI, Fecha
Reserva
#Instalación,Nombre
#Instalacion,Fecha
#DNI

Esquema externo del DFD1

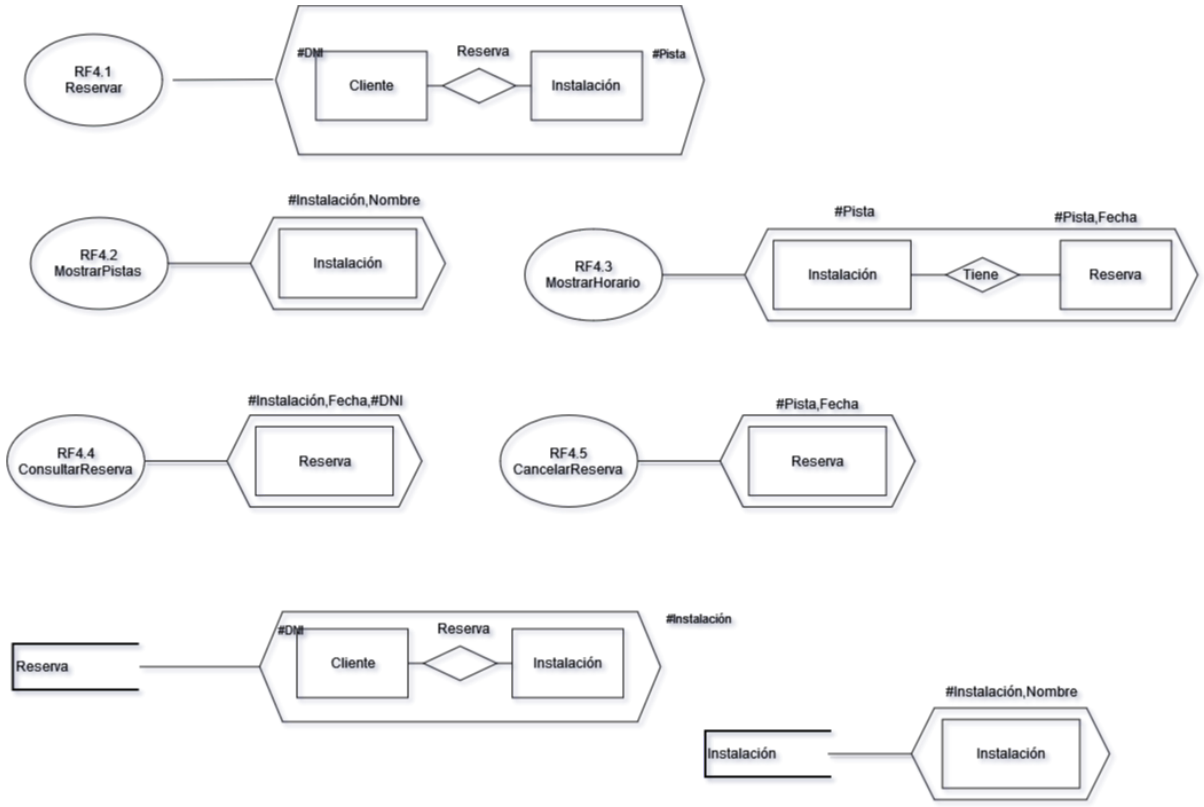
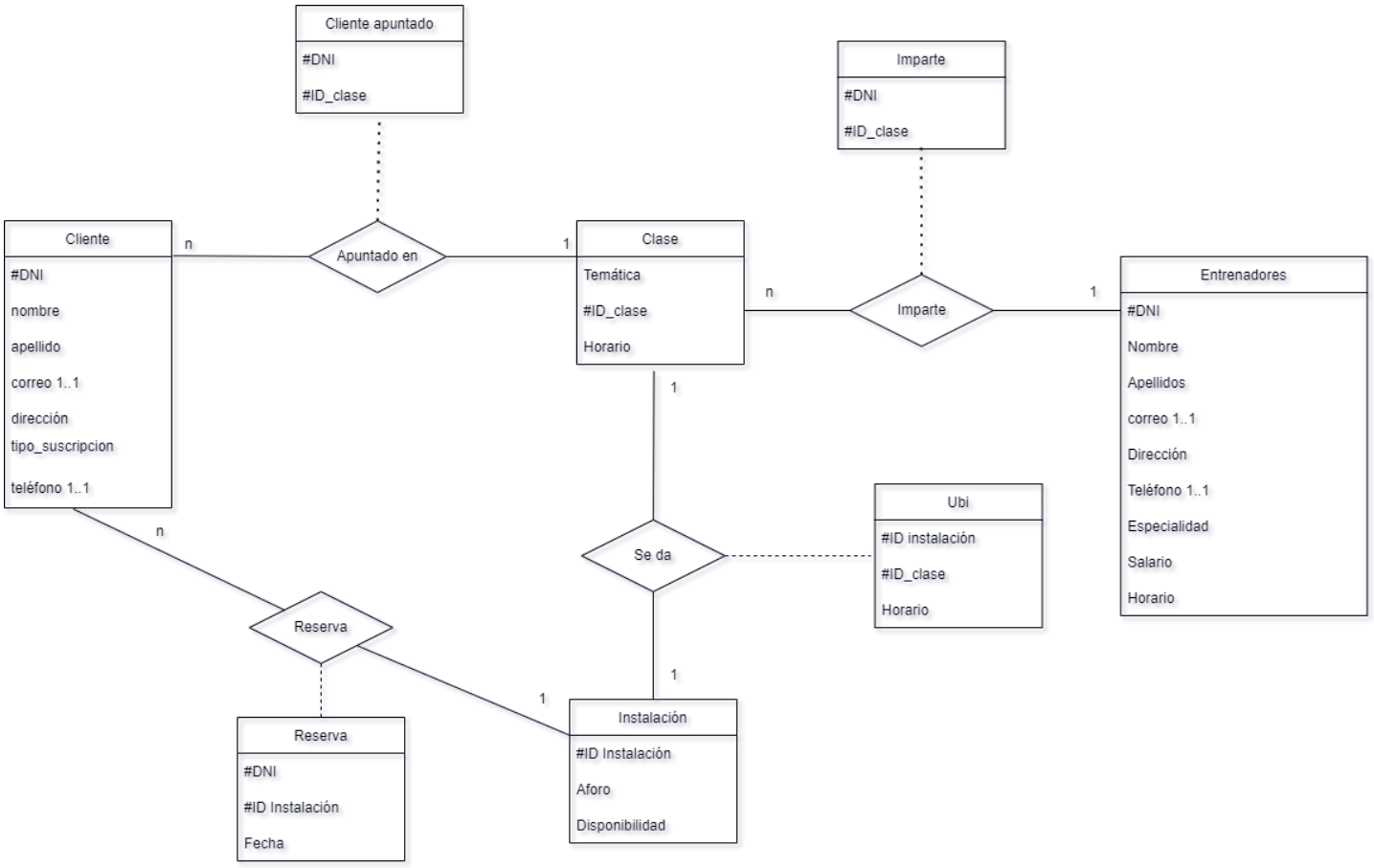


Diagrama ER-UML de la BBDD



5. Sentencias de creación de tablas.

Tablas de clientes

```
CREATE TABLE CLIENTES(
            DNI VARCHAR2(9),
            NOMBRE VARCHAR2(40),
            APELLIDOS VARCHAR2(40),
            CORREO VARCHAR2(60),
            DIRECCION VARCHAR2(40),
            TELEFONO NUMBER(9),
            TIPO_SUSCRIPCION VARCHAR2(9),
            CLASES_APUNTADAS NUMBER(1),
            CONSTRAINT CK_MAX_CLASES CHECK (CLASES_APUNTADAS >=0 AND
CLASES_APUNTADAS <= 25),
            CONSTRAINT PK_CLIENTES PRIMARY KEY (DNI),
            CONSTRAINT UK_CLIENTES_CORREO UNIQUE (CORREO),
            CONSTRAINT UK_CLIENTES_TELEFONO UNIQUE (TELEFONO),
            CONSTRAINT CK_CLIENTES CHECK (TIPO_SUSCRIPCION IN
('NORMAL', 'MEDIO', 'PREMIUM')))
```

```
CREATE TABLE APUNTADO(
            DNI VARCHAR2(9),
            ID_CLASE VARCHAR2(9),
            CONSTRAINT FK_APUNTADO FOREIGN KEY (DNI) REFERENCES
CLIENTES (DNI) ON DELETE CASCADE,
            CONSTRAINT FK_APUNTADO_CLASE FOREIGN KEY (ID_CLASE)
REFERENCES CLASE (ID_CLASE) ON DELETE CASCADE,
            CONSTRAINT PK_APUNTADO PRIMARY KEY(DNI,ID_CLASE)
            )
```

Tablas de entrenadores

```
CREATE TABLE ENTRENADORES(
            DNI VARCHAR2(9),
            NOMBRE VARCHAR2(20),
            APELLIDOS VARCHAR2(40),
            CORREO VARCHAR2(60),
            DIRECCION VARCHAR2(60),
            TELEFONO NUMBER(9),
            ESPECIALIDAD VARCHAR2(30) CHECK (ESPECIALIDAD='Raqueta' OR
ESPECIALIDAD='Equipo' OR ESPECIALIDAD='Personal') ,
            SALARIO NUMBER DEFAULT 0,
            CONSTRAINT PK_ENTRENADORES PRIMARY KEY (DNI),
            CONSTRAINT UK_ENTRENADORES_CORREO UNIQUE (CORREO),
            CONSTRAINT UK_ENTRENADORES_TELEFONO UNIQUE (TELEFONO))
```

Tablas de clases

```
CREATE TABLE CLASE(
            id_clase VARCHAR2(9),
            tematica VARCHAR2(20),
            horario  DATE,
            CONSTRAINT PK_CLASE PRIMARY KEY(id_clase))
```

```
CREATE TABLE IMPARTE(
            DNI VARCHAR2(9),
            id_clase VARCHAR2(9),
            CONSTRAINT EK_CLASEI FOREIGN KEY (id_clase) REFERENCES
CLASE (id_clase) ON DELETE CASCADE,
            CONSTRAINT EK_DNI FOREIGN KEY (DNI) REFERENCES ENTRENADORES
(DNI) ON DELETE CASCADE,
            CONSTRAINT PK_IMPARTE PRIMARY KEY(id_clase,DNI))
```

```
CREATE TABLE LUGAR(
            id_instalacion VARCHAR2(9),
            id_clase VARCHAR2(9),
            CONSTRAINT EK_CLASEL FOREIGN KEY (id_clase) REFERENCES
CLASE (id_clase) ON DELETE CASCADE,
            CONSTRAINT EK_INSTALACION FOREIGN KEY (id_instalacion)
REFERENCES INSTALACION (id_instalacion) ON DELETE CASCADE,
            CONSTRAINT PK_LUGAR PRIMARY KEY(id_clase,id_instalacion))
```

Tablas de instalación

```
CREATE TABLE INSTALACION(
            id_instalacion VARCHAR2(9),
            aforo NUMBER,
            CONSTRAINT IPK_CLASE PRIMARY KEY(id_instalacion))
```

```
CREATE TABLE RESERVA(
            DNI VARCHAR2(9),
            ID_INSTALACION VARCHAR2(9),
            FECHA DATE,
            CONSTRAINT PK_RESERVA PRIMARY KEY(DNI),
            CONSTRAINT FK_RESERVA_INSTALACION FOREIGN
KEY(ID_INSTALACION) REFERENCES INSTALACION)
```

```
CREATE TABLE RESERVA_HISTORICO(
             DNI VARCHAR2(9),
             ID_INSTALACION VARCHAR2(9),
             FECHA DATE,
             CONSTRAINT PK_RESERVA_HIST PRIMARY KEY(DNI),
             CONSTRAINT FK_RESERVA_INSTALACION FOREIGN
KEY(ID_INSTALACION) REFERENCES INSTALACION)
```

6. Descripción de transacciones identificadas

Transacciones de clientes

Para clientes, las transacciones identificadas son las siguientes:

1. Creación de un cliente en la BBDD:
   Se debe controlar que el cliente con un DNI dado, no aparezca ya en la BBDD.
   Se debe controlar que los campos introducidos sean correctos.
   Una vez se hayan introducido todos los datos correctamente, se dará de alta el cliente.

2. Modificación de un cliente:
   Se debe controlar que el cliente con un DNI dado, aparezca en la BBDD.
   Se debe controlar que los campos introducidos sean correctos.
   Una vez se hayan introducido todos los datos correctamente, se modificará el cliente.

3. Borrado de un cliente:
   Se debe controlar que el cliente con un DNI dado, aparezca en la BBDD.
   Se debe controlar que los campos introducidos sean correctos.
   Una vez se hayan introducido todos los datos correctamente, se dará de baja el cliente.

4. Apuntar un cliente a una clase:
   Se debe controlar que el cliente con un DNI dado, aparezca en la BBDD.
   Se debe controlar que la clase con un ID dado, aparezca en la BBDD.
   Se debe controlar que esa clase tenga un instalación asignada.
   Se debe controlar que la instalación para esa clase no supere el aforo establecido.
   Se debe controlar que los campos introducidos sean correctos.
   Una vez se hayan introducido todos los datos correctamente, se dará de alta el cliente en la clase.

Transacciones de entrenadores

1. Creación de un entrenador:
   Se requiere control sobre el DNI, que no aparezca en la base de datos.
   Se requiere control sobre el formato de los campos.
2. Borrado de un entrenador:
   Se requiere control sobre el DNI, que ya exista en la base de datos.
3. Cálculo del salario de un entrenador:
   Se requiere control sobre el DNI, que ya exista en la base de datos.
4. Devolución del horario de un entrenador:
   Se requiere control sobre el DNI, que ya exista en la base de datos.
5. Mostrar los entrenadores existentes

Transacciones de clases

1. Crear una clase:
   Inserción de la clase, no requiere control ya que a priori puede haber clases de todo tipo.
2. Borrar una clase:
   Se comprueba que haya clases creadas.
   Se comprueba que los valores introducidos sean correctos.
3. Mostrar clases:
   Comprueba que haya clases y si las hay las muestra. No se requiere control de ningún tipo.
4. Asignar entrenador a clase:
   Se comprueba que haya clases creadas.
   Se comprueba que haya entrenadores.
   Se comprueba que haya clases que no tengan entrenadores asignados.
   Se comprueba que los datos introducidos sean correctos.
5. Asignar instalación:
   Se comprueba que haya clases creadas.
   Se comprueba que haya instalaciones.
   Se comprueba que haya clases que no tengan instalación asignada.
   Se comprueba que los datos introducidos sean correctos.

Transacciones de instalaciones

1. Registrar Instalación
   Inserción de la instalación, no requiere control
2. Mostrar Instalaciones
   Se comprueba que haya Instalaciones creadas
3. Reservar una Instalación
   Comprueba que el id_instalación está asignado a alguna instalación
4. Cancelar una Reserva
   Se comprueba que haya reservas.
   Se comprueba que los valores introducidos sean correctos.
5. Mostrar Reservas de una instalación en un dia
   Se comprueba que haya reservas.
6. Consultar Reserva
   Se comprueba que haya reservas.

7. Código de los disparadores implementados.

Disparadores de clientes

```
CREATE OR REPLACE TRIGGER CONTROL_TELEFONO
        BEFORE INSERT OR UPDATE ON CLIENTES FOR EACH ROW
        WHEN (NEW.TELEFONO = -1)
        BEGIN
            raise_application_error(-20601,'Ha introducido un carácter
alfabético en el campo teléfono');
        END;
```

Debido a la implementación de cómo se añade y modifica un cliente, si este mete un carácter alfabético, el campo del teléfono se pondrá a -1. Por tanto a la hora de realizar el INSERT se comprobará si este vale -1. Si se da el caso, saltará el tigger con el mensaje. En caso contrario la ejecución continuará con normalidad.

```
CREATE OR REPLACE TRIGGER CONTROL_LONGITUD_TELEFONO
        BEFORE INSERT OR UPDATE ON CLIENTES
        FOR EACH ROW
        WHEN (NEW.TELEFONO NOT LIKE '_________')
        BEGIN
            raise_application_error(-20604,'Longitud de teléfono no
válida');
        END;
```

El trigger comprueba que la longitud del teléfono sea de 9 dígitos.
A la hora de la inserción y modificación se comprobarán ambos triggers.

```
CREATE OR REPLACE TRIGGER CONTROL_CORREO
        BEFORE INSERT OR UPDATE ON CLIENTES FOR EACH ROW
        WHEN (NEW.CORREO NOT LIKE '_%@__%.__%')
        BEGIN
            raise_application_error(-20600,'Ha introducido un carácter
alfabético en el campo teléfono');
        END;
```

El trigger comprueba que el formato del correo electrónico sea del tipo “”@””.””

```
CREATE OR REPLACE TRIGGER CONTROL_SUSCRIPCION
        BEFORE UPDATE OF TIPO_SUSCRIPCION
        ON CLIENTES FOR EACH ROW
        WHEN (NEW.TIPO_SUSCRIPCION NOT IN ('NORMAL', 'MEDIO',
'PREMIUM'))
        BEGIN
            raise_application_error(-20602,'Se ha introducido un tipo
de suscripción no válido');
        END
```

El trigger comprueba que a la hora de insertar o modificar un cliente, el tipo de suscripción sea únicamente NORMAL, MEDIO o PREMIUM.

```
CREATE OR REPLACE TRIGGER CONTROL_DNI
            BEFORE INSERT OR UPDATE OF DNI
            ON CLIENTES FOR EACH ROW
            DECLARE
                LETRA VARCHAR2(1);
            BEGIN
                SELECT SUBSTR(:NEW.DNI,9,9) INTO LETRA FROM DUAL;
                IF (:NEW.DNI NOT LIKE '_________') THEN
                    raise_application_error(-20603,'Formato de DNI
incorrecto');
                ELSE
                    IF LETRA NOT IN ('A' ,'B', 'C', 'D', 'E', 'F', 'G',
'H', 'I', 'J', 'K', 'L', 'M', 'N', 'Ñ', 'O', 'P', 'Q', 'R', 'S', 'T',
'W', 'X', 'Y', 'Z') THEN
                        raise_application_error(-20603,'Formato de DNI
incorrecto');
                    END IF;
                END IF;
            END;
```

El trigger comprueba dos cosas, primero, que la longitud de este sea de 9 dígitos y segundo, que los 8 primeros dígitos sean numéricos y que el último sea alfabético.

```
CREATE OR REPLACE TRIGGER CONTROL_CLASES_APUNTADAS
        AFTER UPDATE OF CLASES_APUNTADAS
        ON CLIENTES FOR EACH ROW
        BEGIN
            IF(:NEW.TIPO_SUSCRIPCION = 'NORMAL') THEN
                IF(:NEW.CLASES_APUNTADAS >= 6) THEN
                    raise_application_error(-20605, 'Número de máximo
de clases alcanzada');
                END IF;
            ELSIF (:NEW.TIPO_SUSCRIPCION = 'MEDIO') THEN
                IF (:NEW.CLASES_APUNTADAS >= 16) THEN
                    raise_application_error(-20606, 'Número de máximo
de clases alcanzada');
                END IF;
            ELSIF (:NEW.TIPO_SUSCRIPCION = 'PREMIUM') THEN
                IF (:NEW.CLASES_APUNTADAS >= 26) THEN
                    raise_application_error(-20607, 'Número de máximo
de clases alcanzada');
                END IF;
            END IF;
        END;
```

El trigger comprueba que después de apuntar un cliente a clase, este no sobrepase su límite de clases a las que se puede apuntar. Si esto ocurre, se lanzará el trigger y saltará una excepción en el programa, indicando la razón del error.

Disparadores de entrenadores

```
CREATE OR REPLACE TRIGGER existe
            BEFORE INSERT ON ENTRENADORES
            FOR EACH ROW
            DECLARE
                cuantos NUMBER(1):=0;
                aux_dni ENTRENADORES.DNI%TYPE;
            BEGIN
                SELECT COUNT(*) INTO cuantos FROM ENTRENADORES
                WHERE DNI = :new.DNI;
                IF (cuantos>0) THEN
                    SELECT DNI INTO aux_dni FROM ENTRENADORES WHERE
DNI=:NEW.DNI;
                    RAISE_APPLICATION_ERROR(-20601,aux_dni||' ese DNI
ya existe en la base de datos');
                END IF;
            END;
```

Dado que el DNI es clave primario no puede existir en la base de datos dos entrenadores con el mismo DNI. Este trigger comprueba que no existe el DNI que se quiere introducir no está ya en la base de datos. Si está salta con un mensaje de error.

```
CREATE OR REPLACE TRIGGER formatoDNI
            BEFORE INSERT OR UPDATE OF DNI ON ENTRENADORES
            FOR EACH ROW
            DECLARE
                LETRA VARCHAR2(1);
            BEGIN
                SELECT SUBSTR(:new.DNI,9,9) INTO LETRA FROM DUAL;
                IF (:new.DNI NOT LIKE '_________') THEN
                     raise_application_error(-20603,'El DNI ha de tener
9 caracteres');
                ELSE
                    IF LETRA NOT IN ('A' ,'B', 'C', 'D', 'E', 'F', 'G',
'H', 'I', 'J', 'K', 'M', 'N', 'Ñ', 'O', 'P', 'Q', 'R', 'S', 'T', 'W',
'X', 'Y', 'Z') THEN
                                raise_application_error(-20603,'El ultimo
caracter del DNI ha de ser una letra');
                    END IF;
                END IF;
            END;
```

El DNI (español) ha de tener 9 caracteres siendo el último de ellos una letra. Si alguna de estas dos condiciones se cumple entonces salta el trigger con un mensaje de error, uno para que tenga los 9 caracteres y otro si no cumple que el último de ellos sea una letra.

```
CREATE OR REPLACE TRIGGER largotlf
            BEFORE INSERT ON ENTRENADORES
            FOR EACH ROW
            WHEN (new.telefono NOT LIKE '_________')
            BEGIN
                 raise_application_error(-20601,'El telefono ha de tener
9 digitos');
            END;
```

Al igual que en el trigger anterior, el teléfono ha de tener 9 dígitos. Si esto no se cumple el trigger salta con su correspondiente mensaje de error.

Disparadores de clases

```
       triggerInstalacion =  """
           create or replace trigger NO_OCUPADA
           before insert
           ON LUGAR
           FOR EACH ROW

           DECLARE
               v_horario_instalacion  clase.horario%TYPE;
               fechaAux clase.horario%TYPE;



           BEGIN
               select horario INTO fechaAux from clase where
id_clase=:new.id_clase;

               FOR I IN(SELECT * FROM LUGAR WHERE
id_instalacion=:new.id_instalacion)
                   LOOP
                   SELECT HORARIO into v_horario_instalacion FROM CLASE
WHERE id_clase=I.id_clase;
                   IF (v_horario_instalacion=fechaAux) then

raise_application_error(-20600,:new.id_instalacion ||' La instalación
está ocupada a la en el horario de esa clase.');
                   END IF;
                   END LOOP;
           END;
       """
```

Este trigger comprueba que la instalación no esté ocupado a la hora de la clase que se le intenta asignar.

```
        triggerEntrenador =  """
            create or replace trigger NO_OCUPADO
            before insert
            ON IMPARTE
            FOR EACH ROW

            DECLARE
                v_horario_entrenador  clase.horario%TYPE;
                fechaAux clase.horario%TYPE;


            BEGIN
                select horario INTO fechaAux from clase where
id_clase=:new.id_clase;

                FOR I IN(SELECT * FROM IMPARTE WHERE dni=:new.dni)
                    LOOP
                    SELECT HORARIO into v_horario_entrenador FROM CLASE
WHERE id_clase=I.id_clase;
                    IF (v_horario_entrenador=fechaAux) then
                            raise_application_error(-20600,:new.dni ||'
El entrenador está ocupado en el horario de esa clase.');
                    END IF;
                    END LOOP;
            END;
        """
```

Este trigger comprueba que el entrenador no esté ocupado a la hora de la clase que se le intenta asignar.

Disparadores de instalación

```
CREATE TRIGGER trigger_reserva_historico
        AFTER INSERT ON reserva
        FOR EACH ROW
        BEGIN
        INSERT INTO reserva_historico(dni,id_instalacion, fecha)
        VALUES (NEW.dni, NEW.id_instalacion, NEW.fecha);
        END;
```

Este trigger permite tener un histórico de las reservas para futuras comprobaciones, haciendo que cada vez que ocurra un Insert en Reserva también se inserte en Reserva_Historico.

8. Elección del software

Decidimos implementar el SI con Python el primer día de clases. El motivo fue que uno de los integrantes tenía conocimientos previos acerca del lenguaje y comentó que este no era muy complicado y se podía aprender bastante rápido debido a su parecido con C++.

Aunque los demás no sabíamos casi nada de Python optamos por este, para poder así aprender un lenguaje nuevo, que a día de hoy es bastante solicitado en el mercado laboral y tiene ciertas características que otros lenguajes no tienen.

Otro motivo fue la fácil accesibilidad para conectar con el SGBD, ya que solo necesitamos descargar el driver ODBC desde la página de Devart para Windows. En caso de Linux también tenemos la opción de descargar el driver desde la página de Devart e instalarlo vía terminal.

Por otro lado, intentamos realizar una interfaz gráfica mediante TKinter pero debido a la carga de trabajo por el resto de asignaturas, finalmente decidimos no llevarla a cabo, por ese motivo, solo está implementado el main y el subsistema de clientes.

Dirección del repositorio por si el zip no funciona.