PROYECTO

GIMNASIO CENTURY FITNESS

3º ENTREGABLE

Miembros del proyecto:

FRANCISCO JAVIER MACERO VÁZQUEZ

JOSÉ CALCEDO VÁZQUEZ

FRAN ZÁJARA GÓMEZ

ANTONIO JESÚS RODRÍGUEZ MORENO

ÍNDICE

1. INTRODUCCIÓN AL PROBLEMA

Este proyecto surge a partir de la página web del gimnasio Century fitness de Sevilla, una página web diseñada para facilitar a los clientes por medio de recursos tecnológicos, la visualización de sus instalaciones, horarios, precios y diferentes clases de las que dispondrán.

Sin embargo, dicha página web puede resultar insuficiente, pues no dispone de las herramientas necesarias para satisfacer las demandas del cliente.

Tras el estudio del dominio del problema, las necesidades a satisfacer, la situación actual y tras hablar con varias personas que son clientes de dicho gimnasio, llegamos a la conclusión de que es necesario una herramienta web que permita crear una estructura de soporte más eficaz que embarque muchas más posibilidades a la hora de la gestión del gimnasio.

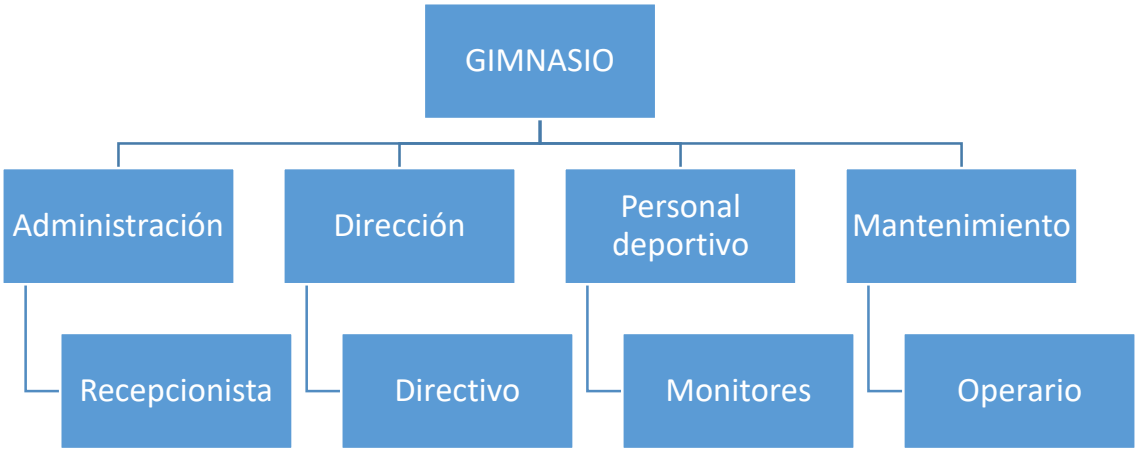
Por ello, el objetivo principal de este proyecto es la creación de dicha herramienta web que complemente a la actual, además de satisfacer nuevas necesidades que han surgido a lo largo del tiempo, como por ejemplo la necesidad de herramientas que permitan la gestión de reservas de clases online, control personal de un entrenamiento y dieta específica, además de un foro en el que los clientes puedan interactuar con los monitores.

1.1. MAPA DE LOCALIZACIONES

Situación del Century Fitness según Google Maps

El gimnasio Century Fitness se encuentra cerca del Polígono Norte, en Sevilla. Es el único establecimiento a nivel regional de Century Fitness.

1.2. ORGANIGRAMA



2.GLOSARIO DE TÉRMINOS

Actividades	Diferentes clases que se imparten en las instalaciones.
Cliente	Persona que contrata los servicios del gimnasio.
Dieta	Serie de alimentos que consume un cliente.
Horario	Tiempo determinado en el que se imparten las diferentes actividades.
Identificación	Método por el cual se puede comprobar si una persona es cliente.
Instalaciones	Lugar de uso y disfrute de los clientes.
Mantenimiento	Tiempo dedicado por las personas encargadas de que el gimnasio se mantenga limpio y ordenado fuera del horario donde los clientes disfrutan de las instalaciones.
Matricula	suscripción al gimnasio por tiempo determinado.
Meta	Objetivo marcado por cada cliente.
Monitores	Personas encargadas de impartir clases y asesorar los entrenamientos.
Ofertas	Promociones de matrículas.
Producto	Artículo deportivo que se puede comprar en el gimnasio
Recepción	Lugar de gestión del gimnasio
Reserva	Petición de un cliente para obtener un determinado servicio del gimnasio
Rutina	Serie de ejercicios físicos y dieta a realizar por el cliente según su meta establecida.

3. CATÁLOGO DE REQUISITOS

3.1. MAPA DE HISTORIAS DE USUARIO

HISTORIAS DE USUARIO						
Gestión de Matriculación		Gestión de Clases	Gestión de Personal		Gestión de Ventas	
RI-001- INFORMACIÓN SOBRE LA MATRÍCULA	RI-002- INFORMACIÓN SOBRE RESERVAS DE CLASES	RI-003- INFORMACIÓN SOBRE LA S CLASES	RI-004- INFORMACIÓN SOBRE LAS RUTINAS	RI-005- INFORMACIÓN SOBRE EL PERSONAL	RI-006- INFORMACIÓN SOBRE LOS PRODUCTOS	RI-001- INFORMACIÓN SOBRE LA VENTA DE PRODUCTOS
RN-001 – Uso exclusivo del gimnasio a clientes matriculados RN-002 – Pérdida del derecho al uso del gimnasio al darse de baja RN-007 – Pago de la matrícula RN-008 – Información sobre clientes	RN-003 - Máximo de reservas por clases RN-004 – Máximo de reservas por cliente	RN-005 – Solo los monitores pueden impartir las clases		RN-009 – Inscripciones de un monitor	RN-006 – Gestión del stock de los productos del almacén	
RF-001 – Matriculación de un cliente		RF-006 – Horarios de las clases	RF-004 – Listado de clientes que solicitan una rutina RF-005 – Registro de rutina de cliente	RF-002 - Garantía de un monitor asignado a una clase		RF-003 – Informe de artículos vendidos
RNF-001 – Control de acceso	RNF-002 – Visualización y funcionamiento correcto			RNF-003 – Tiempo de respuesta		RNF-004 – Fácil de usar

3.2. REQUISITOS GENERALES / OBJETIVOS

RG.-001-Gestión de la matriculación
Como encargado de la administración, quiero una herramienta web, para que el cliente pueda matricularse de forma rápida y sencilla.

RG.-002-Gestión de las clases
Como monitor, quiero un horario fijo adecuado, para impartir las clases que desean los clientes.

RG.-003-Gestión de las Rutinas
Como cliente, quiero obtener una dieta y entrenamiento personalizado, para conseguir lograr mi meta impuesta.

RG.-004-Gestión del personal
Como director, quiero un registro de mi personal, para evitar confusiones.

RG.-005-Gestión de las ventas
Como recepcionista, quiero saber el stock de los productos, para poder venderlos a los clientes.

3.3. REQUISITOS DE INFORMACIÓN

RI-001 – INFORMACIÓN SOBRE LA MATRICULA
Como Recepcionista, Quiero disponer de la información asociada a la matricula: datos personales, pagos realizados y oferta elegida por el cliente Para poder tener control sobre la gestión de matrícula

RI-002 – INFORMACIÓN SOBRE RESERVAS DE CLASES
Como recepcionista, Quiero disponer de la información asociada a las reservas de clases: nombre y clases elegidas Para poder llevar el control de la disponibilidad de las clases

RI-003 – INFORMACIÓN SOBRE LAS CLASES
Como Monitor, Quiero disponer de una lista de los clientes que van a acudir a mi clase y el horario que tendrá Para poder llevar el control y la preparación de la clase a impartir

RI-004 – INFORMACIÓN SOBRE LAS RUTINAS
Como Monitor, Quiero disponer de la información del cliente que solicita la rutina Para poder realizar una dieta y rutina de ejercicios

RI-005 – INFORMACIÓN SOBRE EL PERSONAL
Como Director, Quiero disponer de la información sobre el personal: nombre, fecha de inicio, fecha final y cargo que ocupa Para poder llevar un control del personal que dispongo

RI-006 – INFORMACIÓN SOBRE LOS PRODUCTOS
Como Recepcionista, Quiero disponer de la siguiente información sobre catálogos de productos: nombre, precio y unidades de las que dispongo Para poder recomendar a los clientes los mejores productos

RI-007 – INFORMACIÓN SOBRE LAS VENTAS DE PRODUCTOS
Como Director, Quiero disponer de la siguiente información sobre las ventas de productos: productos en stock y cantidad de cada producto vendida Para poder realizar las compras de cada producto antes de que se terminen y saber qué cantidad debo comprar

RI-008 – INFORMACIÓN SOBRE LAS NÓMINAS DE LOS EMPLEADOS
Como Director, Quiero que mi sistema tenga información detallada de mis empleados Para saber cuándo cobran

3.4. REGLAS DE NEGOCIO

RN-001 – Uso exclusivo del gimnasio a clientes matriculados
Como director, Quiero que los clientes no matriculados no puedan usar el gimnasio o acudir a las actividades realizadas Para evitar confusiones a la hora de identificar a los clientes.

RN-002 – Pérdida del derecho al uso del gimnasio al darse de baja
Como director, Quiero que los clientes que se den de baja no puedan usar el gimnasio o acudir a sus actividades Para evitar clientes morosos.

RN-003 - Máximo de reservas por clases
Como director, Quiero que no se supere el número máximo de participantes por cada clase impartida Para evitar el exceso de clientes en la sala.

RN-004 – Máximo de reservas por cliente
Como director, Quiero que el cliente no pueda reservar más de una clase a la misma hora Para evitar reservas innecesarias.

RN-005 – Solo los monitores pueden impartir las clases
Como director, Quiero que el monitor asociado a una clase sea el único que pueda impartir dicha clase Para evitar confusiones entre monitores

RN-006 – Gestión del stock de los productos del almacén
Como director, Quiero que cuando la cantidad del producto que hay en stock sea su mínimo correspondiente se avise al director. Para que este realice un nuevo encargo al proveedor de los productos.

RN-007 – Pago de la matrícula
Como director, Quiero que el cliente realice el pago en metálico o con tarjeta en la recepción antes de que finalice el mes Para que no se dé de baja su matrícula.

RN-008 – Información sobre clientes
Como director, quiero tener disponible la siguiente información sobre los clientes de mi empresa: NIF, nombre y apellidos, dirección, teléfonos (fijos y/o móviles) y correo electrónico Para evitar duplicar los datos de un cliente.

RN-009 – Inscripciones de un monitor
Como director, Quiero que un monitor no pueda inscribirse en dos o más clases a la misma hora Para evitar que una clase se quede sin monitor.

3.5. REQUISITOS FUNCIONALES

RF-001 – Matriculación de un cliente
Como recepcionista, Quiero un formulario sobre el cliente Para poder realizar su matriculación.

RF-002 - Garantía de un monitor asignado a una clase
Como director, Quiero que el sistema permita obtener un informe que garantice que al menos un monitor está asignado a una determinada clase Para poder llevar un control de monitores.

RF-003 – Informe de artículos vendidos
Como director, Quiero que el sistema permita obtener un informe que muestre el número de productos vendidos de cada tipo Para tener un control sobre el stock de los productos.

RF-004 – Listado de clientes que solicitan una rutina
Como monitor, Quiero un listado sobre los clientes que solicitan una rutina Para poder elaborar las rutinas correspondientes.

RF-005 – Registro de rutina de cliente
Como monitor, Quiero que el sistema permita poder añadir un registro Para poder asociar la rutina correspondiente al cliente.

RF-006 – Horarios de las clases
Como director, Quiero que el sistema me permita obtener la información de los horarios de cada clase Para llevar un control sobre ellas.

3.6. REQUISITOS NO FUNCIONALES

RNF-001 – Control de acceso
Como director, Quiero sólo puedan acceder al sistema los empleados de mi organización, con el usuario y contraseña dado por el director. Para cumplir con la Ley de Protección de Datos

RNF-002 – Visualización y funcionamiento correcto
Como director, Quiero que el sistema permita visualizarse y funcionar correctamente en cualquier navegador Para que todo usuario pueda hacer disfrute del mismo sistema.

RNF-003 – Tiempo de respuesta
Como director, Quiero que el sistema permita no tardar más de cinco segundos en mostrar los resultados de una búsqueda, en el caso de que se supere este plazo, el sistema detiene la búsqueda y muestra los resultados encontrados Para acelerar el proceso de interacción entre usuario y sistema.

RNF-004 – Fácil de usar
Como director, Quiero que el sistema permita un fácil y sencillo manejo para cualquier tipo de usuario con o sin conocimientos técnicos Para una comodidad de entendimiento del sistema para el personal.

3.7. PRUEBAS DE ACEPTACIÓN

RN-001 – Uso exclusivo del gimnasio a clientes matriculados
Como director, Quiero que los clientes no matriculados no puedan usar el gimnasio o acudir a las actividades realizadas Para evitar confusiones a la hora de identificar a los clientes.

PRUEBAS DE ACEPTACIÓN:

Un cliente matriculado tiene derecho al uso y disfrute de las instalaciones del gimnasio y actividades impartidas dentro de él.

Un cliente no matriculado no dispondrá de acceso a las instalaciones del gimnasio, ni ningún servicio del mismo.

Un cliente el cual ha finalizado la matriculación no dispondrá de acceso a las instalaciones del gimnasio, ni ningún servicio del mismo.

RN-002 – Pérdida del derecho al uso del gimnasio al darse de baja
Como director, Quiero que los clientes que se den de baja no puedan usar el gimnasio o acudir a sus actividades Para evitar clientes morosos.

PRUEBAS DE ACEPTACIÓN:

Un cliente el cual se ha dado de baja no puede utilizar las instalaciones del gimnasio, ni realizar las actividades impartidas dentro de él.

Un cliente dado de baja no podrá utilizar las instalaciones del gimnasio ni realizar las actividades impartidas dentro de él.

RN-003 - Máximo de reservas por clases
Como director, Quiero que no se supere el número máximo de participantes por cada clase impartida Para evitar el exceso de clientes en la sala.

PRUEBAS DE ACEPTACIÓN:

Un cliente reserva una clase la cual tiene 19 reservas realizadas donde su máximo es 20 y el programa lo permite.

Un cliente reserva una clase la cual tiene 21 reservas realizada donde su máximo es 20 y el programa la deniega.

RN-004 – Máximo de reservas por cliente
Como director, Quiero que el cliente no pueda reservar más de una clase a la misma hora Para evitar reservas innecesarias.

PRUEBAS DE ACEPTACIÓN:

Un cliente que aún no ha realizado ninguna reserva de clase, se dispone a realizar una reserva y el programa lo permite.

Un cliente que ha realizado una reserva, se dispone a realizar otra a la misma hora y el programa la deniega.

RN-005 – Solo los monitores pueden impartir las clases
Como director, Quiero que el monitor asociado a una clase sea el único que pueda impartir dicha clase Para evitar confusiones entre monitores

PRUEBAS DE ACEPTACIÓN:

Un monitor se inscribe en una clase en la que está contratado para impartirla y el programa la deniega.

Un cliente o cualquier persona ajena al gimnasio se inscribe para impartir una clase y el programa la deniega.

RN-006 – Gestión del stock de los productos del almacén
Como director, Quiero que cuando queden una cierta cantidad de productos en el almacén se avise al director Para que este realice un nuevo encargo al proveedor de los productos.

PRUEBAS DE ACEPTACIÓN:

Un cliente compra un producto y al actualizarse el stock queda la cantidad mínima de ese producto, el programa avisa al director.

Un cliente compra un producto y al actualizarse el stock queda una cantidad por encima de la mínima de ese producto, el programa no avisa al director.

RN-007 – Pago de la matrícula
Como director, Quiero que el cliente realice el pago en metálico o en tarjeta en la recepción antes de que finalice el mes Para que no se dé de baja su matrícula tras finalizar el mes.

PRUEBAS DE ACEPTACIÓN:

Un cliente paga antes de la finalización de su matrícula, el programa no da de baja su matrícula cuando el mes finaliza.

Un cliente no paga antes de la finalización de su matrícula, el programa da de baja su matrícula cuando el mes finaliza.

RN-008 – Información sobre clientes
Como director, Quiero tener disponible la siguiente información sobre los clientes de mi empresa: NIF, nombre y apellidos, dirección, teléfonos y correo electrónico. Para evitar duplicar los datos de un cliente

PRUEBAS DE ACEPTACIÓN:

Se registra un cliente nuevo, se pide un listado de clientes y aparece el cliente nuevo.

Se modifican los datos de un cliente, se pide un listado de clientes y aparece el cliente con los datos modificados.

Se elimina un cliente, se pide un listado de clientes y aparece el cliente como eliminado.

RN-009 – Inscripciones de un monitor
Como director, No quiero que un monitor pueda inscribirse en dos o más clases a la misma hora. Para evitar que una clase se quede sin monitor.

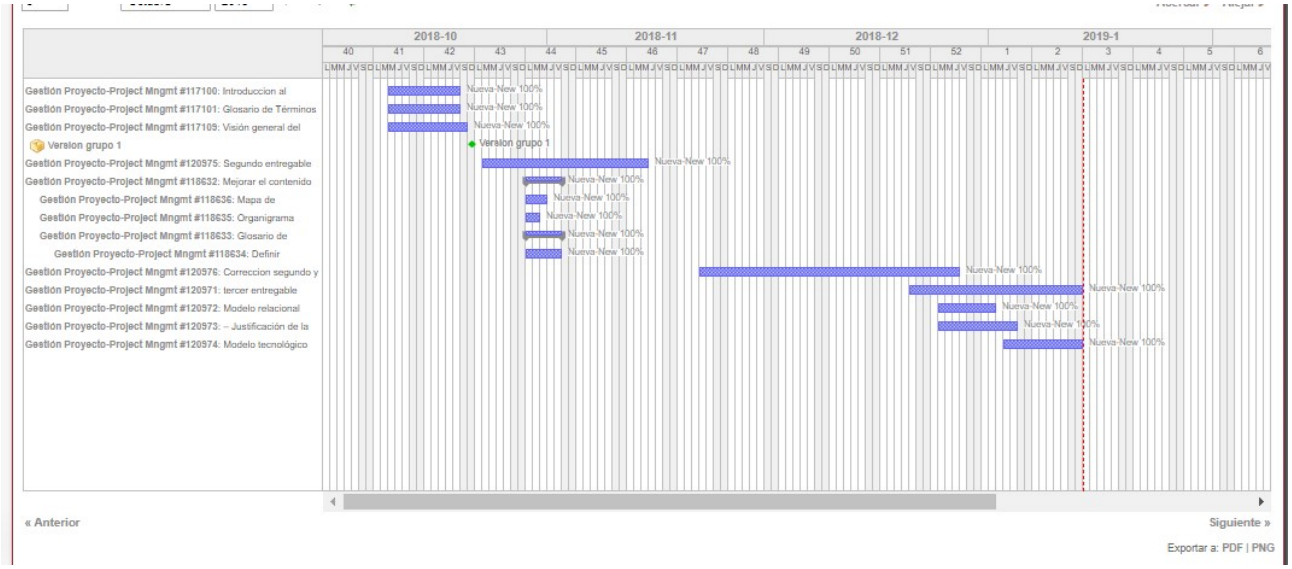
PRUEBAS DE ACEPTACIÓN:

Un monitor se inscribe en una clase y el programa lo permite.

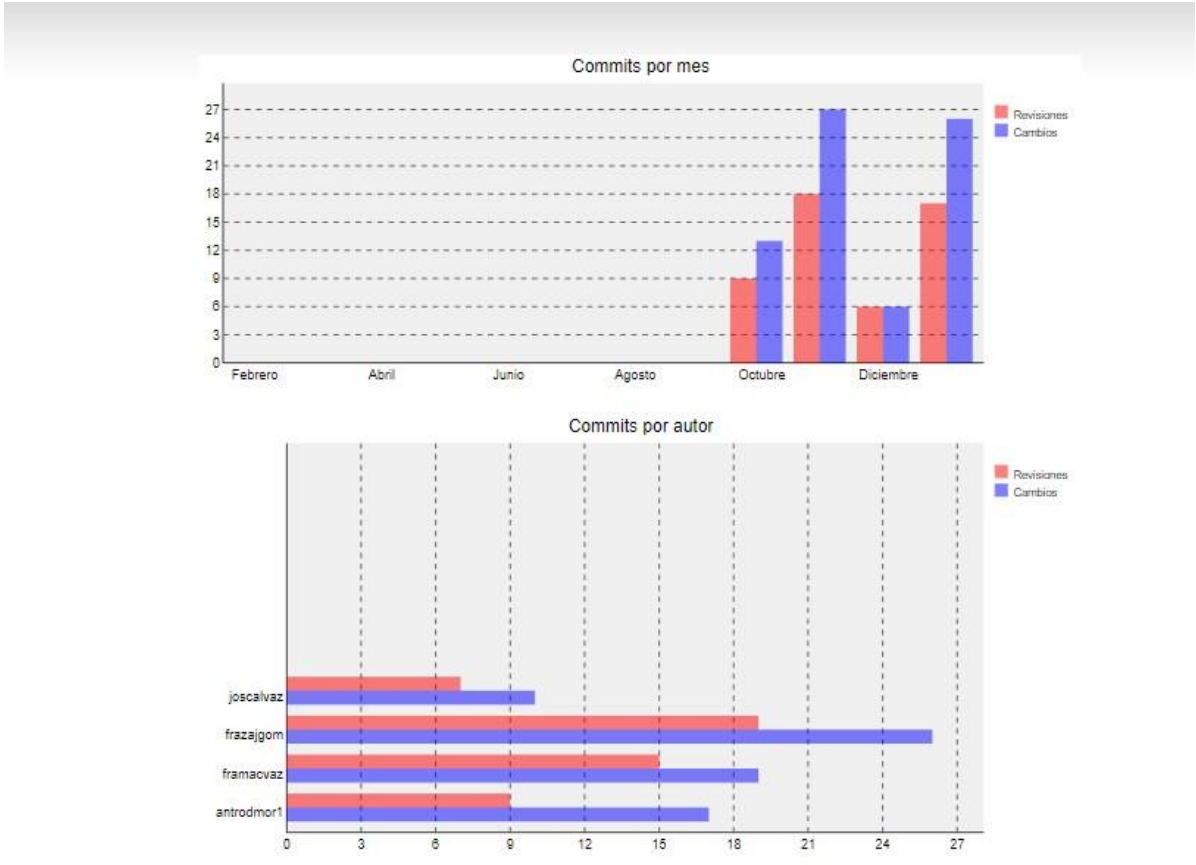
Un monitor se inscribe en una clase y estaba inscrito en otra a la misma hora, el programa lo deniega.

4. ANEXO

4.1. GESTIÓN DEL PROYECTO CON PROJETSII



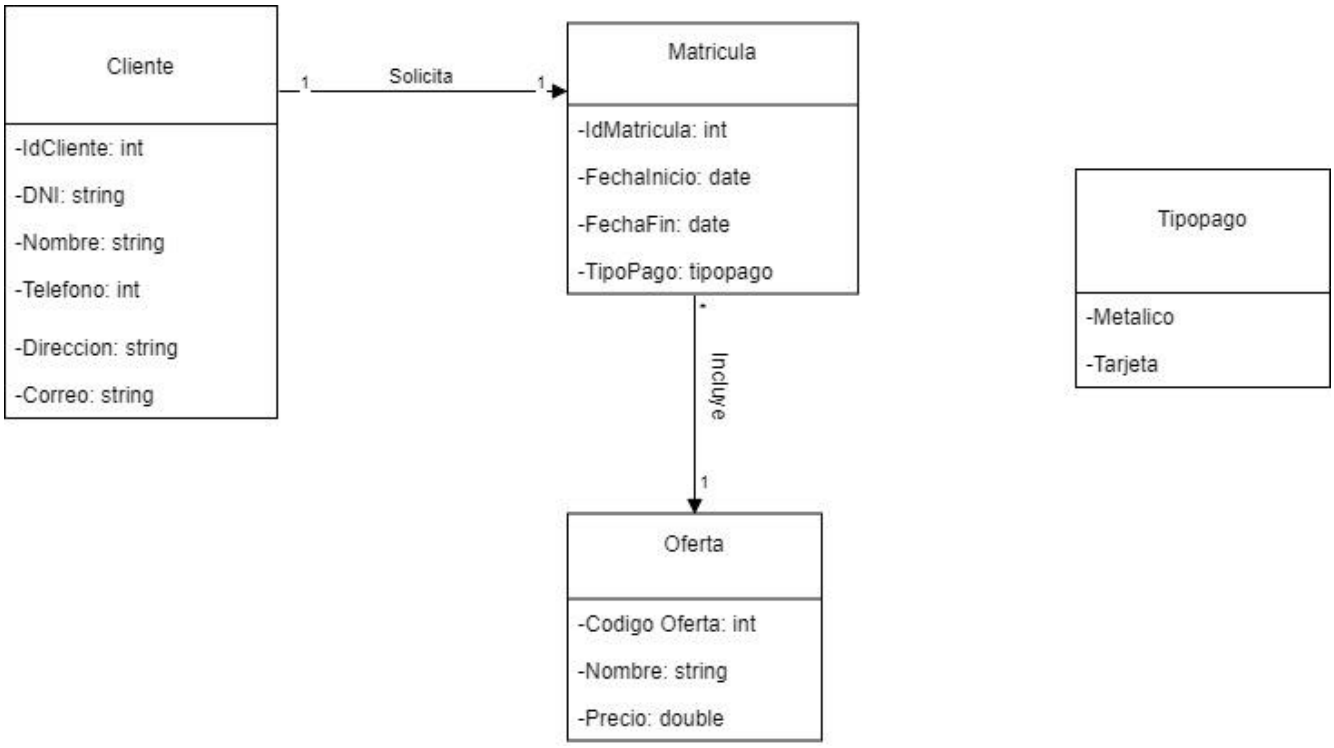
4.2. UTILIZACIÓN DEL REPOSITORIO SVN



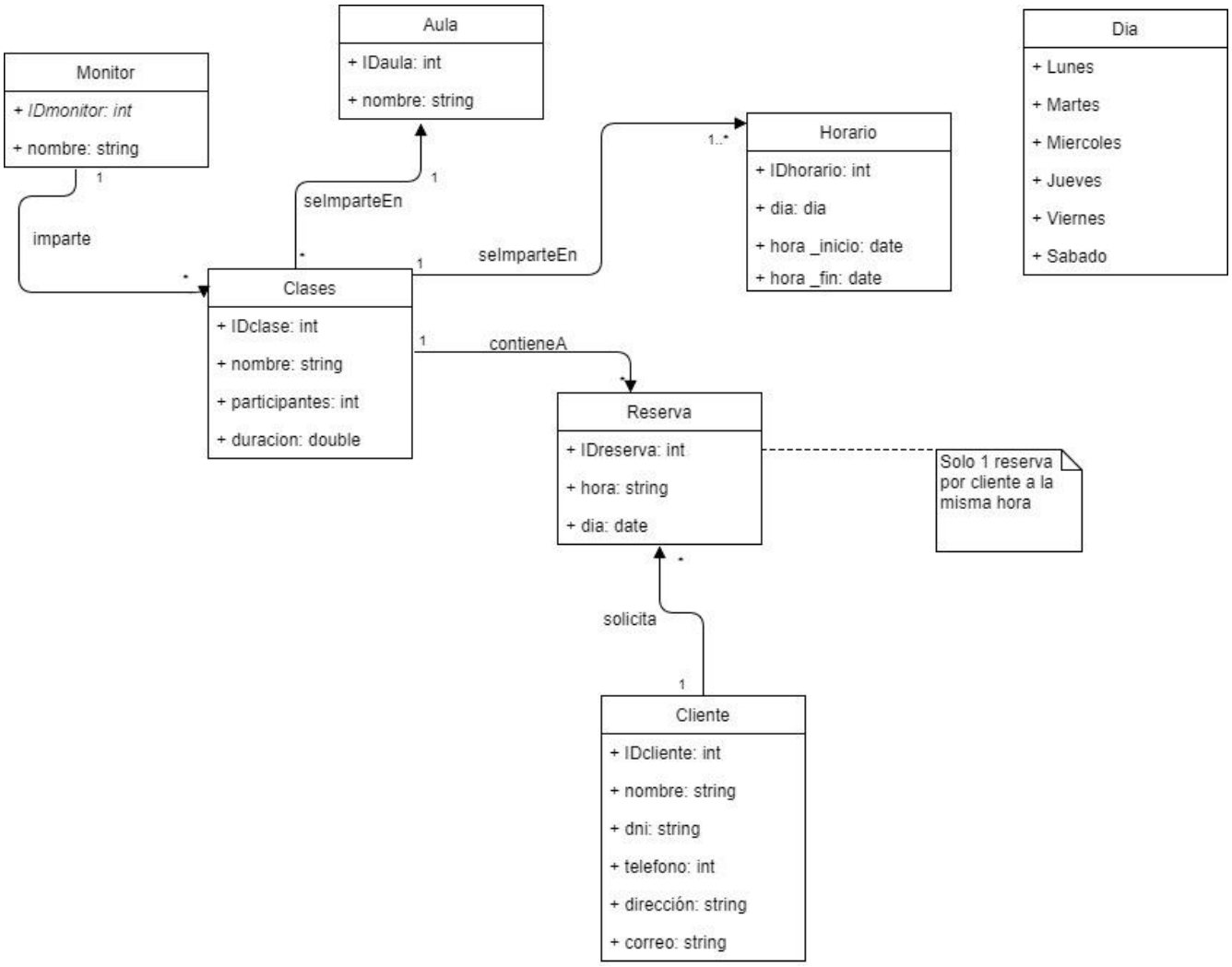
5. MODELO CONCEPTUAL

5.1. DIAGRAMAS DE CLASES UML

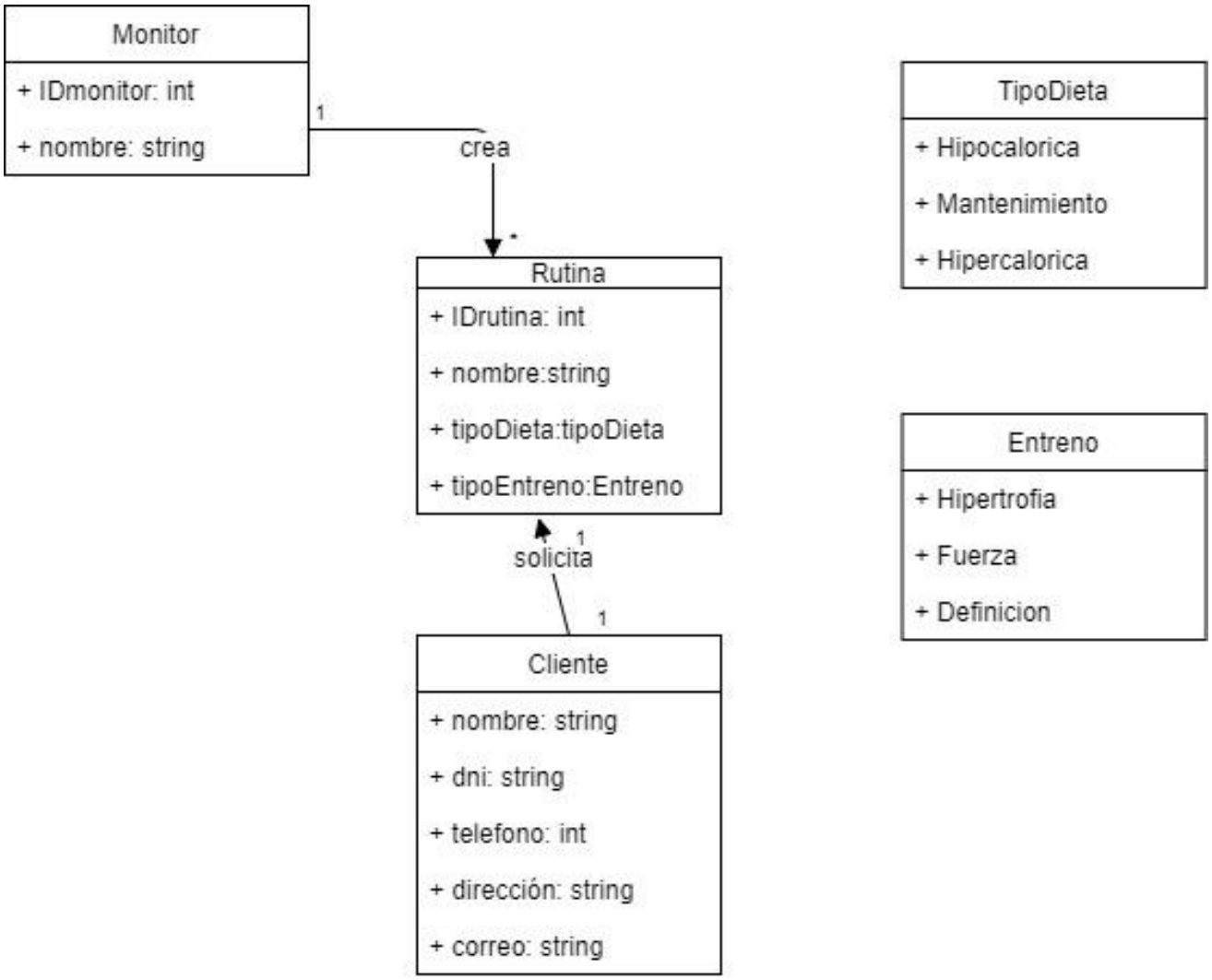
5.1.1. DIAGRAMA DE CLASES DE LA GESTIÓN DE LA MATRICULACIÓN



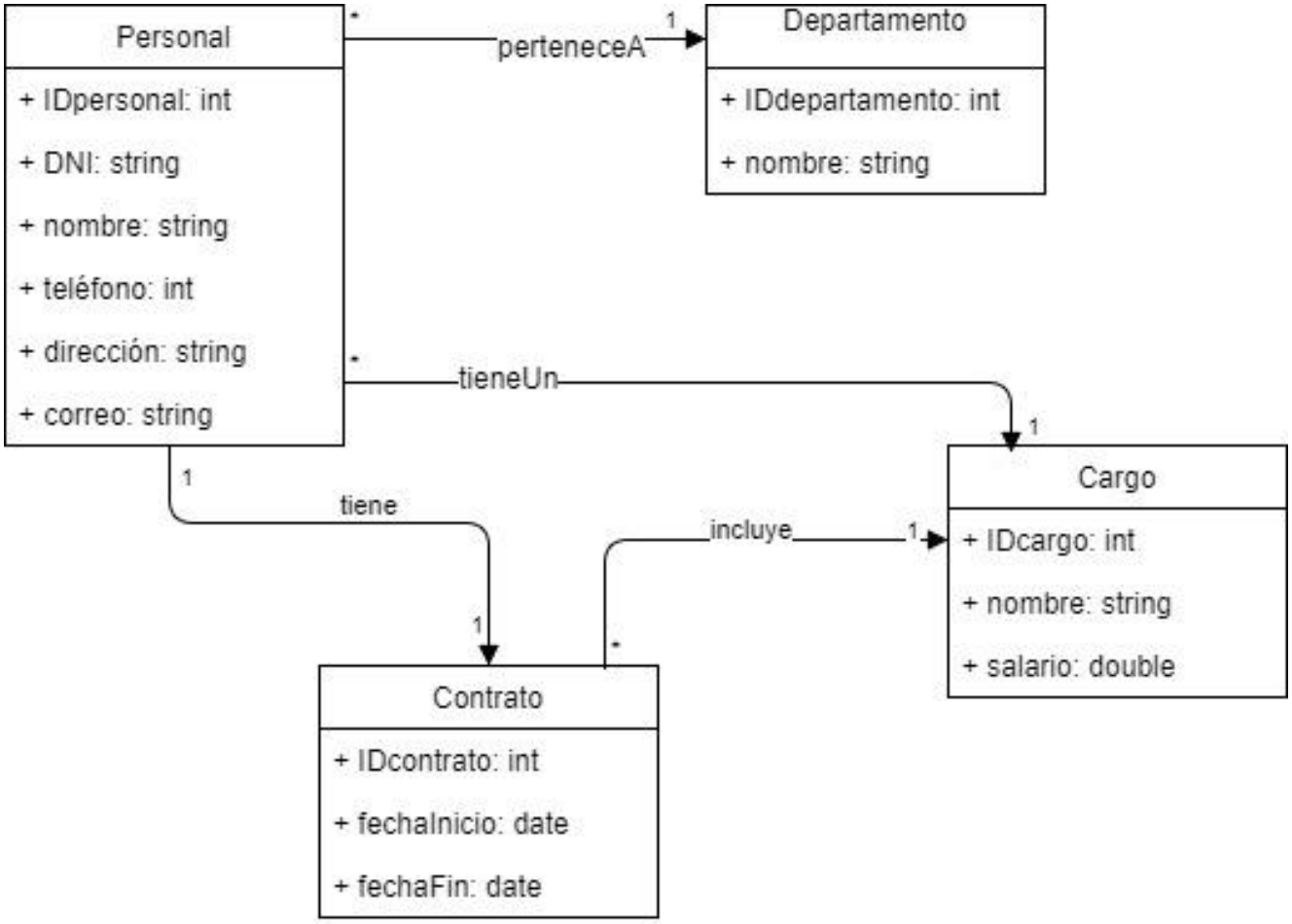
5.1.2. DIAGRAMA DE CLASES DE LA GESTIÓN DE CLASES



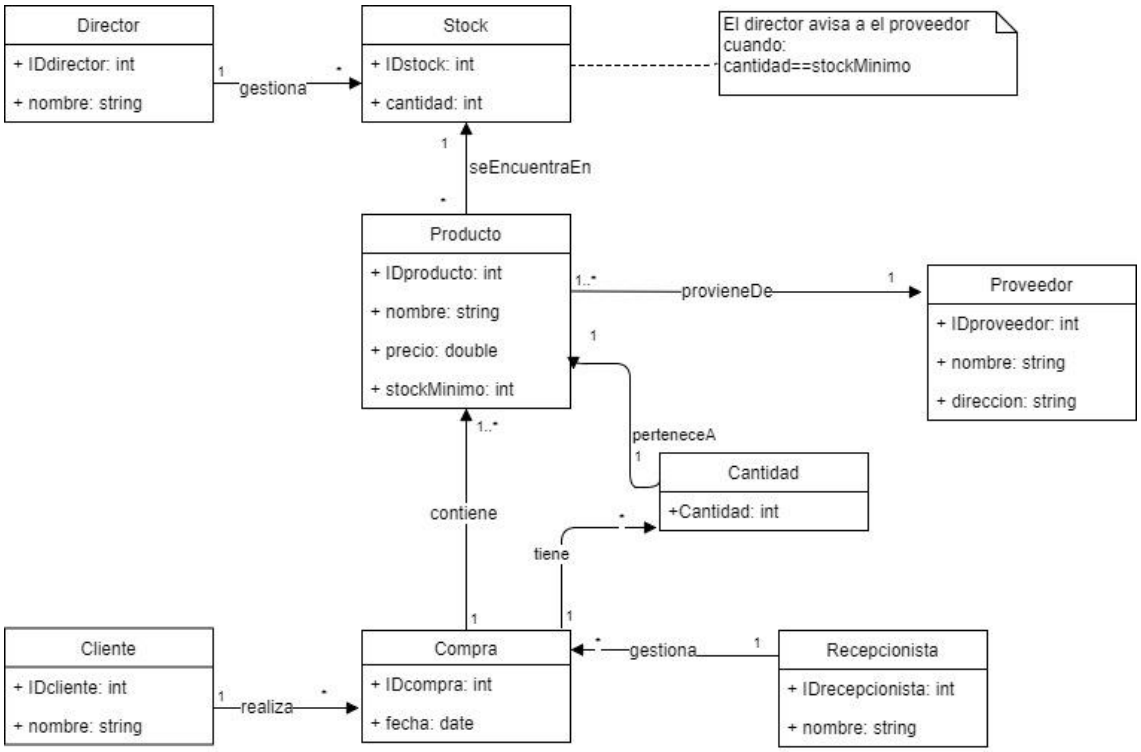
5.1.3. DIAGRAMA DE CLASES DE LA GESTIÓN DE RUTINAS



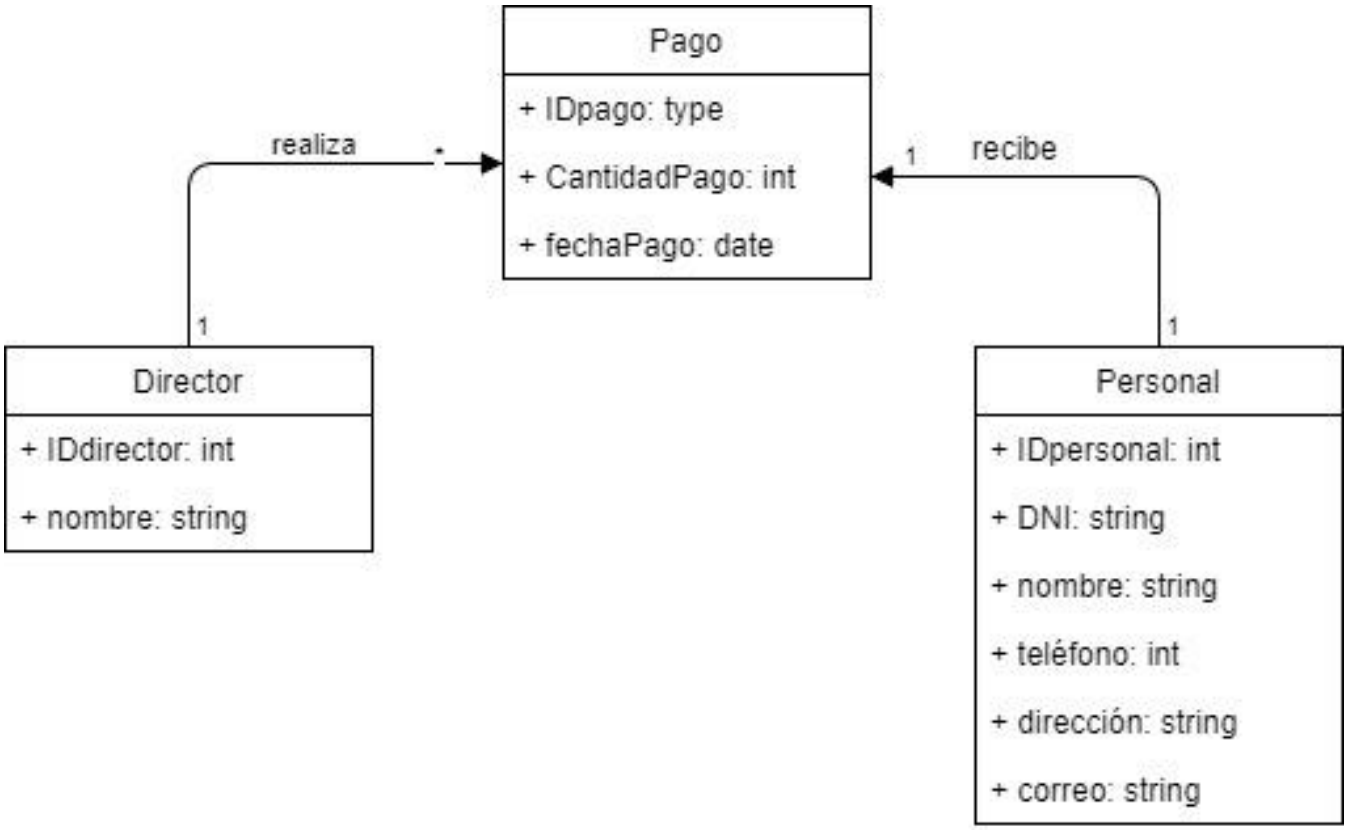
5.1.4. DIAGRAMA DE CLASES DE LA GESTIÓN DEL PERSONAL



5.1.5. DIAGRAMA DE CLASES DE LA GESTIÓN DE VENTAS

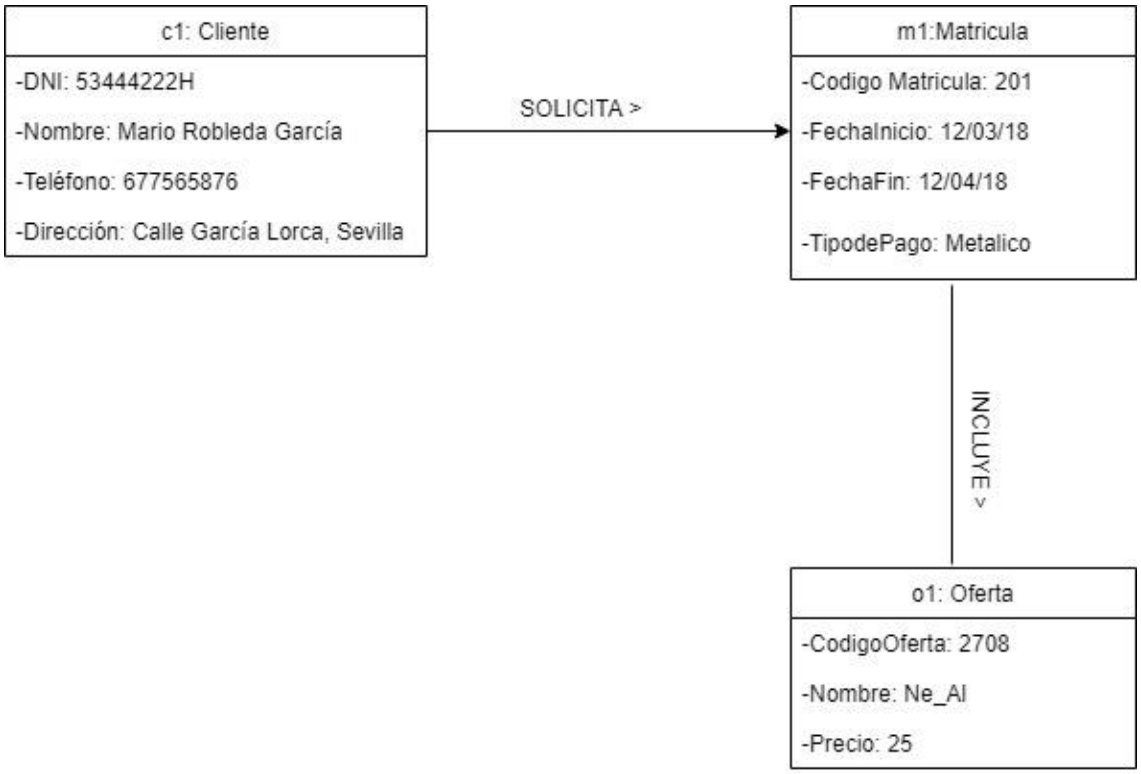


5.1.6. DIAGRAMA DE CLASES DE LA GESTIÓN DE PAGOS

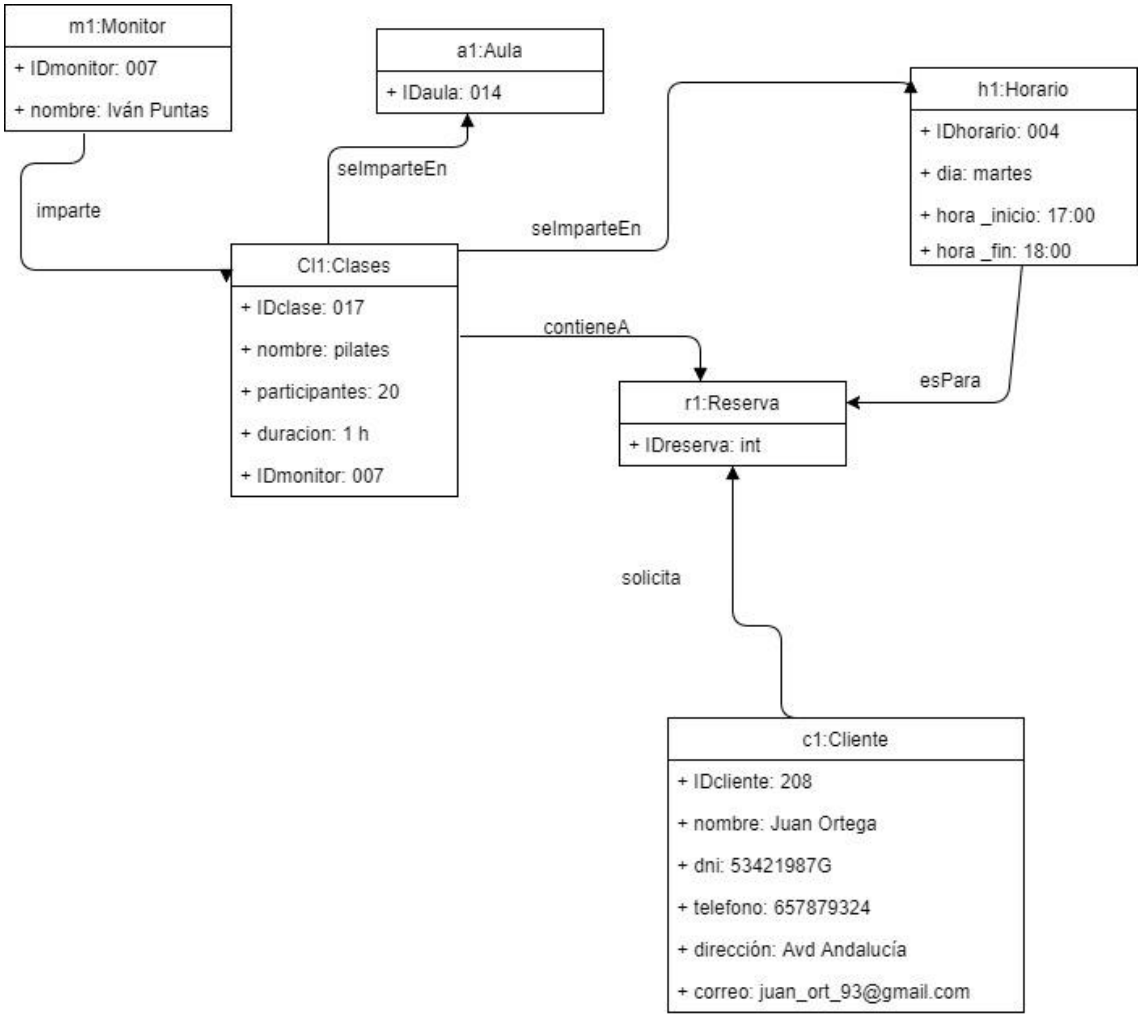


5.2. ESCENARIOS DE PRUEBA

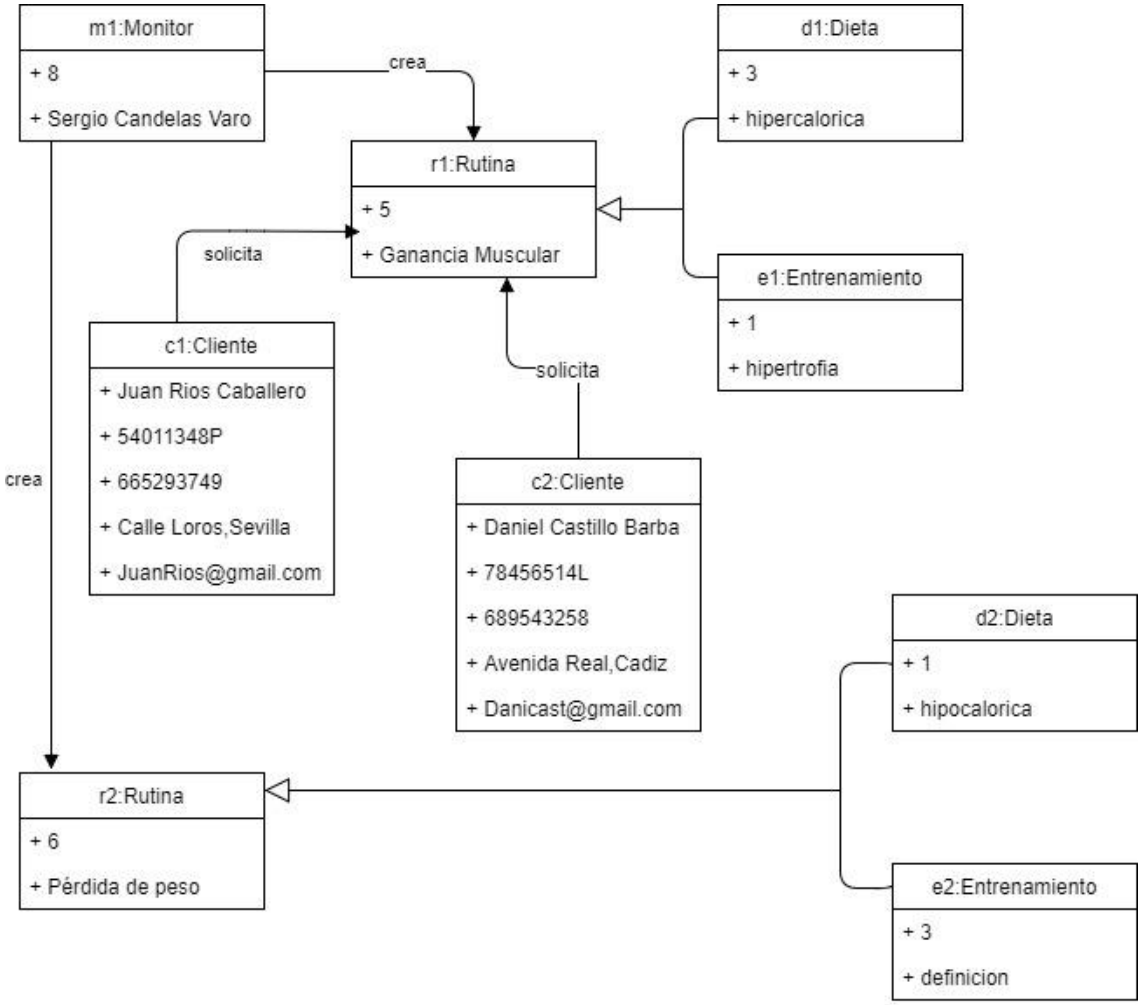
5.2.1. ESCENARIO DE PRUEBA DE LA MATRICULACIÓN



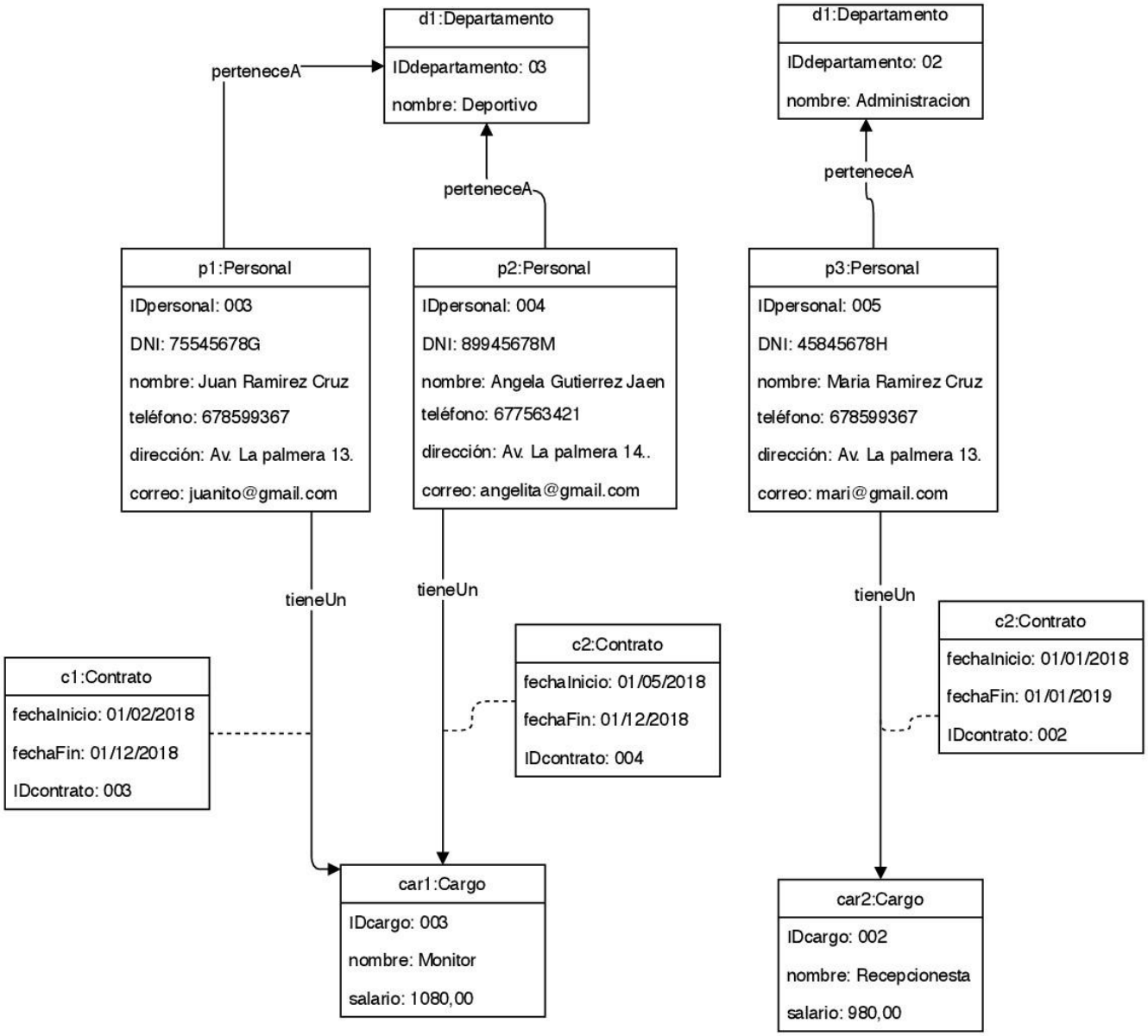
5.2.2. ESCENARIO DE PRUEBA DE LAS CLASES



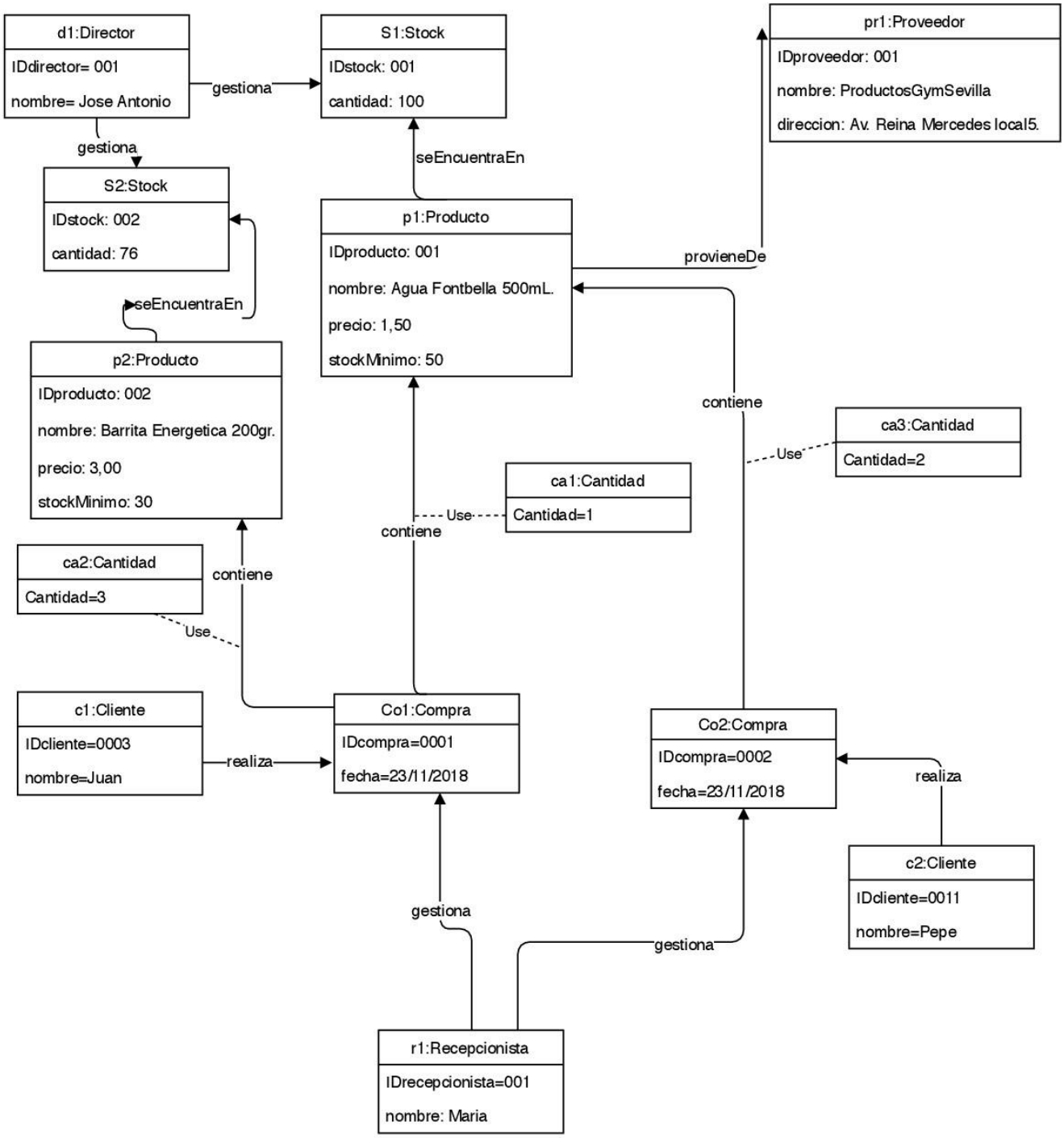
5.2.3. ESCENARIO DE PRUEBA DE LAS RUTINAS



5.2.4. ESCENARIO DE PRUEBA DEL PERSONAL



5.2.5. ESCENARIO DE PRUEBA DE VENTAS



5.3. MATRICES DE TRAZABILIDAD

5.3.1. MATRIZ DE TRAZABILIDAD DE CLASES -REQUISITOS DE INFORMACIÓN

Req. De inf. Clases	RI-001 – INFORMACIÓN SOBRE LA MATRICULA	RI-002 – INFORMACIÓN SOBRE RESERVAS DE CLASES	RI-003 – INFORMACIÓN SOBRE LAS CLASES	RI-004 – INFORMACIÓN SOBRE LAS RUTINAS	RI-005 – INFORMACIÓN SOBRE EL PERSONAL	RI-006 – INFORMACIÓN SOBRE LOS PRODUCTOS	RI-007 – INFORMACIÓN SOBRE LAS VENTAS DE PRODUCTOS
Actividades		✓	✓				
Cliente							
Horario			✓				
Identificación	✓						
Instalaciones							
Mantenimiento					✓		
Matricula	✓						
Meta				✓			
Monitores					✓		
Recepción							
Rutina				✓			
Oferta	✓						

5.3.2. MATRIZ DE TRAZABILIDAD DE CLASES –REGLAS DE NEGOCIO

Reglas de Neg. Clases	RN-001 – Uso exclusivo del gimnasio a clientes matriculados	RN-002 – Pérdida del derecho al uso del gimnasio al darse de baja	RN-003 - Máximo de reservas por clases	RN-004 – Máximo de reservas por cliente	RN-005 – Solo los monitores pueden impartir las clases	RN-006 – Compras solo de productos en stock	RN-007 – Gestión del stock de los productos del almacén	RN-008 – Pago de la matrícula	RN-009 – Información sobre clientes	RN-010 – Inscripciones de un monitor
Actividades			✓		✓					
Cliente	✓	✓						✓	✓	
Horario				✓						
Identificación	✓								✓	
Instalaciones										
Mantenimiento										
Matricula	✓							✓		
Meta									✓	
Monitores										
Recepción								✓		
Rutina									✓	
Oferta								✓		

5.3.3. MATRIZ DE TRAZABILIDAD DE ASOCIACIONES –REQUISITOS DE INFORMACIÓN

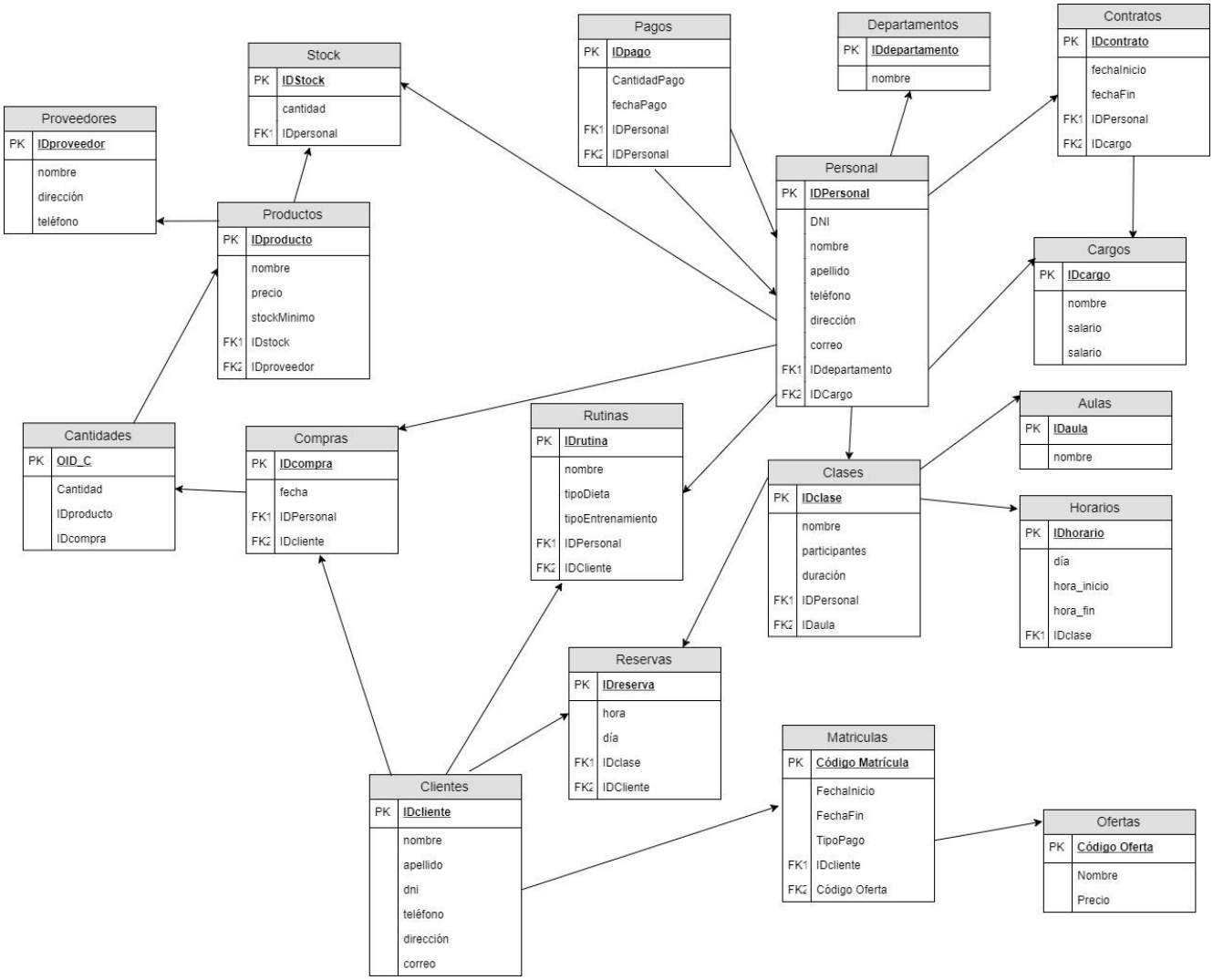
Req. De inf. Asociaciones	RI-001 – INFORMACIÓN SOBRE LA MATRICULA	RI-002 – INFORMACIÓN SOBRE RESERVAS DE CLASES	RI-003 – INFORMACIÓN SOBRE LAS CLASES	RI-004 – INFORMACIÓN SOBRE LAS RUTINAS	RI-005 – INFORMACIÓN SOBRE EL PERSONAL	RI-006 – INFORMACIÓN SOBRE LOS PRODUCTOS	RI-007 – INFORMACIÓN SOBRE LAS VENTAS DE PRODUCTOS
solicita	✓	✓	✓	✓			
incluye	✓						
contieneA		✓	✓			✓	✓
esPara		✓	✓				
seImparteEn		✓	✓				
crea				✓			
perteneceA					✓		
tieneUn					✓		
gestiona						✓	✓
seEncuentraEn						✓	✓
provieneDe						✓	✓
realiza						✓	✓

5.3.4. MATRIZ DE TRAZABILIDAD DE ASOCIACIONES –REGLAS DE NEGOCIO

Reglas de Neg. Asociaciones	RN-001 – Uso exclusivo del gimnasio a clientes matriculados	RN-002 – Pérdida del derecho al uso del gimnasio al darse de baja	RN-003 - Máximo de reservas por clases	RN-004 – Máximo de reservas por cliente	RN-005 – Solo los monitores pueden impartir las clases	RN-006 – Compras solo de productos en stock	RN-007 – Gestión del stock de los productos del almacén	RN-008 – Pago de la matrícula	RN-009 – Información sobre clientes	RN-010 – Inscripciones de un monitor
solicita	✓	✓	✓	✓	✓			✓	✓	✓
incluye	✓	✓						✓	✓	
contieneA			✓	✓		✓	✓			✓
esPara			✓	✓	✓					✓
seImparteEn			✓	✓	✓					✓
crea										
perteneceA										
tieneUn										
gestiona						✓	✓			
seEncuentraEn						✓	✓			
provieneDe						✓	✓			
realiza						✓	✓			

6. MODELO TECNOLÓGICO

6.1. MODELO RELACIONAL EN 3ª FORMA NORMAL



6.2. RELACIONES OBTENIDAS AL APLICAR LA TRANSFORMACIÓN DEL MODELO CONCEPTUAL

Tablas	Atributos	Primary Keys	Alternate Keys	Foreign Keys
Aulas	IDaula	IDaula	-	-
Cantidades	OID_C, Cantidad, IDproducto, IDcompra	OID_C	-	IDproducto/productos IDcompra/compras
Cargos	IDcargo, nombre, salario	IDcargo	nombre	-
Clases	IDclase, nombre, participantes, duración, Idpersonal, IDaula	IDclase	-	IDpersonal/personal IDaula/aula
Clientes	IDcliente, nombre, dni, apellidos, telefono, direccion, correo	IDcliente	nombre	-
Compras	IDcompra, fecha, IDpersonal, IDcliente	IDcompra	-	IDpersonal/personal IDcliente/clientes
Contratos	IDcontratos, fechaInicio, fechaFin, IDpersonal, IDpagos	IDcontratos	-	IDpersonal/personal IDpagos/pagos
Departamento	IDdepartamento, nombre	IDdepartamento	nombre	-
Horarios	IDhorario, dia, hora_inicio, hora_fin, IDclase	IDhorario	-	IDclase/clases
Matriculas	CodigoMatricula, FechaInicio, FechaFin, TipoPago, IDcliente, CodigoOferta	CodigoMatricula	-	IDcliente/clientes CodigoOferta/ofertas
Ofertas	CodigoOferta, nombre, precio	CodigoOferta	nombre	-
Pagos	IDpago, cantidadPago, fechaPago, IDpersonal, IDpersonal	IDpago	-	IDpersonal/personal IDpersonal/personal
Personal	IDpersonal, nombre, apellido, dni, telefono, direccion, correo, IDdepartamento, IDcargo	IDpersonal	nombre	IDdepartamento/depar-tamento IDcargo/cargos
Productos	IDproducto, nombre, precio, stockMinimo, IDstock, IDproveedor	IDproducto	nombre	IDstock/stock IDproveedor/proveed-or
Proveedores	IDproveedor, nombre, apellido, dirección, teléfono	IDproveedor	nombre	-
Reservas	IDreserva, hora, día, IDclase, IDcliente	IDreserva	-	IDclase/clase IDcliente/cliente
Rutinas	IDrutinas, nombre, tipoDieta, tipoEntrenamiento, IDpersonal, IDcliente	IDrutinas	nombre	IDpersonal/personal IDcliente/cliente
Stock	IDstock, cantidad, IDpersonal	IDstock	-	IDpersonal/personal

6.3 ESQUEMA ORACLE

```
DROP TABLE pagos;
DROP TABLE cantidades;
DROP TABLE compras;
DROP TABLE productos;
DROP TABLE stock;
DROP TABLE contratos;
DROP TABLE rutinas;
DROP TABLE reservas;
DROP TABLE horarios;
DROP TABLE clases;
DROP TABLE personal;
DROP TABLE proveedores;
DROP TABLE departamentos;
DROP TABLE cargos;
DROP TABLE aulas;
DROP TABLE matriculas;
DROP TABLE ofertas;
DROP TABLE clientes;

-- Borrado de secuencias
DROP SEQUENCE sec_cli;
DROP SEQUENCE sec_ofe;
DROP SEQUENCE sec_mat;
DROP SEQUENCE sec_aul;
DROP SEQUENCE sec_car;
DROP SEQUENCE sec_dep;
DROP SEQUENCE sec_prov;
DROP SEQUENCE sec_per;
DROP SEQUENCE sec_cla;
DROP SEQUENCE sec_hor;
DROP SEQUENCE sec_res;
DROP SEQUENCE sec_rut;
DROP SEQUENCE sec_con;
DROP SEQUENCE sec_sto;
DROP SEQUENCE sec_prod;
DROP SEQUENCE sec_com;
DROP SEQUENCE sec_can;
DROP SEQUENCE sec_pag;
```

```
/* Creación Tabla de Clientes */
CREATE TABLE clientes (
    idcliente   SMALLINT PRIMARY KEY,
    dni         VARCHAR(9) NOT NULL,
    nombre      VARCHAR(30) NOT NULL,
    apellidos   VARCHAR(30) NOT NULL,
    correo      VARCHAR(30) NOT NULL,
    direccion   VARCHAR(30) NOT NULL,
```

```
    telefono    VARCHAR(14) NOT NULL
);

-- Creación de secuencia para generar PK de tabla de Clientes

CREATE SEQUENCE sec_cli;

-- Creación de Trigger de secuencias para generar PK de tabla de
Clientes

CREATE OR REPLACE TRIGGER clientes_pk BEFORE
    INSERT ON clientes
    FOR EACH ROW
BEGIN
    SELECT
        sec_cli.NEXTVAL
    INTO :new.idcliente
    FROM
        dual;

END;
/

/* Creación Tabla de Ofertas */

CREATE TABLE ofertas (
    idoferta   SMALLINT PRIMARY KEY,
    nombre     VARCHAR(30) NOT NULL,
    precio     NUMBER(6, 2) NOT NULL
);

-- Creación de secuencia para generar PK de tabla de Ofertas

CREATE SEQUENCE sec_ofe;
-- Creación de Trigger de secuencias para generar PK de tabla de
Ofertas

CREATE OR REPLACE TRIGGER ofertas_pk BEFORE
    INSERT ON ofertas
    FOR EACH ROW
BEGIN
    SELECT
        sec_ofe.NEXTVAL
    INTO :new.idoferta
    FROM
        dual;

END;
/


/* Creación Tabla de Matriculas */

CREATE TABLE matriculas (
    idMatricula        SMALLINT PRIMARY KEY,
    fechaInicio           DATE NOT NULL,
```

```
    fechaFin           DATE NOT NULL,


    tipoPago           VARCHAR(15)
                CHECK( tipoPago IN('metalico','tarjeta') ),


    idCliente               SMALLINT NOT NULL,
    idOferta       SMALLINT NOT NULL,
    FOREIGN KEY ( idCliente )
        REFERENCES clientes ON DELETE CASCADE,
    FOREIGN KEY ( idOferta )
        REFERENCES ofertas ON DELETE CASCADE
);

-- Creación de secuencia para generar PK de tabla de Matriculas
CREATE SEQUENCE sec_mat;
-- Creación de Trigger de secuencias para generar PK de tabla de
Matriculas
CREATE OR REPLACE TRIGGER matriculas_PK
BEFORE INSERT ON matriculas
FOR EACH ROW
BEGIN
 SELECT sec_mat.nextval INTO :new.IdMatricula FROM dual;
END;
/

/* Creación Tabla de Aulas */
CREATE TABLE aulas (
     idAulas        SMALLINT PRIMARY KEY,
     nombre     VARCHAR(15) NOT NULL
);

-- Creación de secuencia para generar PK de tabla de Aulas

CREATE SEQUENCE sec_aul;
-- Creación de Trigger de secuencias para generar PK de tabla de
Aulas
CREATE OR REPLACE TRIGGER aulas_pk BEFORE
    INSERT ON aulas
    FOR EACH ROW
BEGIN
    SELECT
        sec_aul.NEXTVAL
    INTO :new.idAulas
    FROM
        dual;

END;
/

/* Creación Tabla de Cargos */
CREATE TABLE cargos (
    idcargos   SMALLINT PRIMARY KEY,
    nombre     VARCHAR(30) NOT NULL,
    salario     NUMBER(6, 2) NOT NULL
```

```
);

-- Creación de secuencia para generar PK de tabla de Cargos

CREATE SEQUENCE sec_car;
-- Creación de Trigger de secuencias para generar PK de tabla de
Cargos

CREATE OR REPLACE TRIGGER cargos_pk BEFORE
    INSERT ON cargos
    FOR EACH ROW
BEGIN
    SELECT
        sec_car.NEXTVAL
    INTO :new.idcargos
    FROM
        dual;

END;
/

/* Creación Tabla de Departamentos */
CREATE TABLE departamentos (
    iddepartamentos   SMALLINT PRIMARY KEY,
    nombre     VARCHAR(30) NOT NULL

);

-- Creación de secuencia para generar PK de tabla de
Departamentos

CREATE SEQUENCE sec_dep;
-- Creación de Trigger de secuencias para generar PK de tabla de
Departamentos
CREATE OR REPLACE TRIGGER departamentos_pk BEFORE
    INSERT ON departamentos
    FOR EACH ROW
BEGIN
    SELECT
        sec_dep.NEXTVAL
    INTO :new.iddepartamentos
    FROM
        dual;

END;
/

/* Creación Tabla de Proveedores */
CREATE TABLE proveedores (
    idProveedor   SMALLINT PRIMARY KEY,
    nombre      VARCHAR(30) NOT NULL,
    correo      VARCHAR(30) NOT NULL,
    direccion   VARCHAR(30) NOT NULL,
    telefono    VARCHAR(14) NOT NULL
);
```

```
-- Creación de secuencia para generar PK de tabla de Proveedores

CREATE SEQUENCE sec_prov;

-- Creación de Trigger de secuencias para generar PK de tabla de
Proveedores

CREATE OR REPLACE TRIGGER proveedores_pk BEFORE
    INSERT ON proveedores
    FOR EACH ROW
BEGIN
    SELECT
        sec_prov.NEXTVAL
    INTO :new.idProveedor
    FROM
        dual;

END;
/

/* Creación Tabla de Personal */
CREATE TABLE personal (
    idPersonal   SMALLINT PRIMARY KEY,
    dni          VARCHAR(9) NOT NULL,
    nombre       VARCHAR(30) NOT NULL,
    apellidos    VARCHAR(30) NOT NULL,
    correo       VARCHAR(30) NOT NULL,
    direccion    VARCHAR(30) NOT NULL,
    telefono     VARCHAR(14) NOT NULL,

    idDepartamento      SMALLINT NOT NULL,
    idCargo             SMALLINT NOT NULL,
    FOREIGN KEY ( idDepartamento )
        REFERENCES Departamentos,
    FOREIGN KEY ( idCargo )
        REFERENCES Cargos
);

-- Creación de secuencia para generar PK de tabla de Personal

CREATE SEQUENCE sec_per;

-- Creación de Trigger de secuencias para generar PK de tabla de
Personal

CREATE OR REPLACE TRIGGER personal_pk BEFORE
    INSERT ON personal
    FOR EACH ROW
BEGIN
    SELECT
        sec_per.NEXTVAL
    INTO :new.idPersonal
    FROM
        dual;

END;
```

```
/

/* Creación Tabla de Clases */
CREATE TABLE clases (
    idClase   SMALLINT PRIMARY KEY,
    nombre      VARCHAR(30) NOT NULL,
    participantes   NUMBER(2) NOT NULL,
    duracion      VARCHAR(30) NOT NULL,

    idPersonal      SMALLINT NOT NULL,
    idAula             SMALLINT NOT NULL,
    FOREIGN KEY ( idPersonal )
        REFERENCES personal,
    FOREIGN KEY ( idAula )
        REFERENCES aulas
);

-- Creación de secuencia para generar PK de tabla de Clases

CREATE SEQUENCE sec_cla;

-- Creación de Trigger de secuencias para generar PK de tabla de
Clases

CREATE OR REPLACE TRIGGER clases_pk BEFORE
    INSERT ON clases
    FOR EACH ROW
BEGIN
    SELECT
        sec_cla.NEXTVAL
    INTO :new.idClase
    FROM
        dual;

END;
/

/* Creación Tabla de Horarios */

CREATE TABLE horarios (
    idHorario           SMALLINT PRIMARY KEY,
      dia          VARCHAR(10)
                CHECK( dia IN('Lunes','Martes',
'Miercoles','Jueves','Viernes','Sabado') ),

    horaInicio          VARCHAR(5) NOT NULL,
    horaFin             VARCHAR(5) NOT NULL,

    idClase             SMALLINT NOT NULL,

    FOREIGN KEY ( idClase )
        REFERENCES clases
);

-- Creación de secuencia para generar PK de tabla de Horarios
CREATE SEQUENCE sec_hor;
```

```
-- Creación de Trigger de secuencias para generar PK de tabla de
Horarios
CREATE OR REPLACE TRIGGER horarios_PK
BEFORE INSERT ON horarios
FOR EACH ROW
BEGIN
 SELECT sec_hor.nextval INTO :new.idHorario FROM dual;
END;
/

/* Creación Tabla de Reservas */

CREATE TABLE reservas (
    idReserva        SMALLINT PRIMARY KEY,
    hora           VARCHAR(5) NOT NULL,
    dia           DATE NOT NULL,


    idClase               SMALLINT NOT NULL,
    idCliente       SMALLINT NOT NULL,
    FOREIGN KEY ( idClase )
        REFERENCES clases ON DELETE CASCADE,
    FOREIGN KEY ( idCliente )
        REFERENCES clientes ON DELETE CASCADE
);

-- Creación de secuencia para generar PK de tabla de Reservas
CREATE SEQUENCE sec_res;
-- Creación de Trigger de secuencias para generar PK de tabla de
reservas
CREATE OR REPLACE TRIGGER reservas_PK
BEFORE INSERT ON reservas
FOR EACH ROW
BEGIN
 SELECT sec_res.nextval INTO :new.idReserva FROM dual;
END;
/


/* Creación Tabla de Rutinas */

CREATE TABLE rutinas (
    IDrutina        SMALLINT PRIMARY KEY,
    nombre             VARCHAR(9)  NOT NULL,

    tipoDieta           VARCHAR(15)
               CHECK( tipoDieta
IN('Hipercalorica','Mantenimiento', 'Hipocalorica') ),
    tipoEntrenamiento           VARCHAR(15)
               CHECK( tipoEntrenamiento
IN('Hipertrofia','Fuerza', 'Definicion') ),


    idPersonal      SMALLINT NOT NULL,
    idCliente       SMALLINT NOT NULL,
    FOREIGN KEY ( idPersonal )
```

```
        REFERENCES personal ON DELETE CASCADE,
    FOREIGN KEY ( idCliente )
        REFERENCES clientes ON DELETE CASCADE
);

-- Creación de secuencia para generar PK de tabla de Rutinas
CREATE SEQUENCE sec_rut;
-- Creación de Trigger de secuencias para generar PK de tabla de
Rutinas
CREATE OR REPLACE TRIGGER rutinas_PK
BEFORE INSERT ON rutinas
FOR EACH ROW
BEGIN
 SELECT sec_rut.nextval INTO :new.IDrutina FROM dual;
END;
/

/* Creación Tabla de Contratos */

CREATE TABLE contratos (
    IDcontrato        SMALLINT PRIMARY KEY,
    fechaInicio           DATE NOT NULL,
    fechaFin           DATE NOT NULL,


    idPersonal      SMALLINT NOT NULL,
    IDcargo       SMALLINT NOT NULL,
    FOREIGN KEY ( idPersonal )
        REFERENCES personal,
    FOREIGN KEY ( IDcargo )
        REFERENCES cargos
);

-- Creación de secuencia para generar PK de tabla de Contratos
CREATE SEQUENCE sec_con;
-- Creación de Trigger de secuencias para generar PK de tabla de
contratos
CREATE OR REPLACE TRIGGER contratos_PK
BEFORE INSERT ON contratos
FOR EACH ROW
BEGIN
 SELECT sec_con.nextval INTO :new.idcontrato FROM dual;
END;
/

/* Creación Tabla de Stock */
CREATE TABLE stock (
    IDstock        SMALLINT PRIMARY KEY,
    cantidad           NUMBER(5,0) NOT NULL,


    idPersonal      SMALLINT NOT NULL,
    FOREIGN KEY ( idPersonal )
        REFERENCES personal
);
```

```
-- Creación de secuencia para generar PK de tabla de Stock
CREATE SEQUENCE sec_sto;
-- Creación de Trigger de secuencias para generar PK de tabla de
Stock
CREATE OR REPLACE TRIGGER stock_PK
BEFORE INSERT ON stock
FOR EACH ROW
BEGIN
 SELECT sec_sto.nextval INTO :new.idStock FROM dual;
END;
/



CREATE TABLE productos (
     idProducto SMALLINT PRIMARY KEY,
     nombre     VARCHAR(30) NOT NULL,
     precio NUMBER(6, 2) NOT NULL,
     stockMinimo int,
     idStock SMALLINT,
     FOREIGN KEY ( idStock )
        REFERENCES stock,
     idProveedor SMALLINT,
     FOREIGN KEY ( idProveedor )
        REFERENCES proveedores
);

-- Creación de secuencia para generar PK de tabla de Productos

CREATE SEQUENCE sec_prod;
-- Creación de Trigger de secuencias para generar PK de tabla de
Productos
CREATE OR REPLACE TRIGGER productos_pk BEFORE
    INSERT ON productos
    FOR EACH ROW
BEGIN
    SELECT
        sec_prod.NEXTVAL
    INTO :new.idProducto
    FROM
        dual;

END;
/

CREATE TABLE compras (
     idCompra SMALLINT PRIMARY KEY,
     fecha DATE,
     idPersonal SMALLINT,
     FOREIGN KEY ( idPersonal )
        REFERENCES personal ON DELETE CASCADE,
     idCliente  SMALLINT,
     FOREIGN KEY ( idCliente )
        REFERENCES clientes ON DELETE CASCADE
);
```

```
-- Creación de secuencia para generar PK de tabla de Compras

CREATE SEQUENCE sec_com;
-- Creación de Trigger de secuencias para generar PK de tabla de
Compras
CREATE OR REPLACE TRIGGER compras_pk BEFORE
    INSERT ON compras
    FOR EACH ROW
BEGIN
    SELECT
        sec_com.NEXTVAL
    INTO :new.idCompra
    FROM
        dual;

END;
/

CREATE TABLE cantidades (
     OID_C SMALLINT PRIMARY KEY,
     cantidad INT,
     idProducto SMALLINT,
     FOREIGN KEY ( idProducto )
        REFERENCES productos,
     idCompra SMALLINT,
     FOREIGN KEY ( idCompra )
        REFERENCES compras
);

-- Creación de secuencia para generar PK de tabla de Compras

CREATE SEQUENCE sec_can;
-- Creación de Trigger de secuencias para generar PK de tabla de
Compras
CREATE OR REPLACE TRIGGER cantidades_pk BEFORE
    INSERT ON cantidades
    FOR EACH ROW
BEGIN
    SELECT
        sec_can.NEXTVAL
    INTO :new.OID_C
    FROM
        dual;

END;
/

CREATE TABLE pagos (
     idPago SMALLINT PRIMARY KEY,
     cantidadPago NUMBER(6, 2) NOT NULL,
     fechaPago DATE,
     idpersonal1 SMALLINT NOT NULL,
     FOREIGN KEY ( idPersonal1 )
        REFERENCES personal,
     idpersonal2 SMALLINT NOT NULL,
```

```
     FOREIGN KEY ( idPersonal2 )
        REFERENCES personal
);

-- Creación de secuencia para generar PK de tabla de pagos

CREATE SEQUENCE sec_pag;
-- Creación de Trigger de secuencias para generar PK de tabla de
Pagos
CREATE OR REPLACE TRIGGER pagos_pk BEFORE
    INSERT ON pagos
    FOR EACH ROW
BEGIN
    SELECT
        sec_pag.NEXTVAL
    INTO :new.idPago
    FROM
        dual;

END;
/


--Procedimientos y funciones asociadas a las reglas funcionales
--RF-001
CREATE OR REPLACE PROCEDURE PR_Formulario_añadir_cliente (v_dni
IN clientes.dni%TYPE, v_nombre IN
clientes.nombre%TYPE,v_apellidos IN clientes.apellidos%TYPE,
v_correo IN clientes.correo%TYPE, v_direccion IN
clientes.direccion%TYPE, v_telefono IN clientes.telefono%TYPE)
IS
BEGIN
INSERT INTO clientes (dni, nombre, apellidos, correo, direccion,
telefono) VALUES (v_dni, v_nombre, v_apellidos, v_correo,
v_direccion, v_telefono);
COMMIT;
END;
/

CREATE OR REPLACE PROCEDURE PR_Formulario_añadir_oferta
(v_nombre IN ofertas.nombre%TYPE, v_precio IN
ofertas.precio%TYPE)
IS
BEGIN
INSERT INTO ofertas (nombre, precio) VALUES (v_nombre,
v_precio);
COMMIT;
END;
/

CREATE OR REPLACE PROCEDURE PR_Form_generar_matricula
(v_fechaInicio IN matriculas.fechaInicio%TYPE, v_fechaFin IN
matriculas.fechaFin%TYPE,
v_tipoPago IN matriculas.tipoPago%TYPE, v_idCliente IN
matriculas.idCliente%TYPE, v_idOferta IN
matriculas.idOferta%TYPE)
```

```
IS
BEGIN
INSERT INTO matriculas (fechaInicio, fechaFin, tipoPago,
idCliente, idOferta) VALUES (v_fechaInicio, v_fechaFin,
v_tipoPago, v_idCliente, v_idOferta);
COMMIT;
END;
/



--RF-002
CREATE OR REPLACE PROCEDURE PR_Mon_Imparte_clase (CURSORMEMORIA
OUT SYS_REFCURSOR)
AS
BEGIN
OPEN CURSORMEMORIA FOR SELECT c.nombre AS nombre_c, p.nombre AS
nombre_p FROM clases C, personal P WHERE C.idpersonal =
P.idpersonal;
END;
/

/* CODIGO DE PRUEBA PARA EJECTUAR UN PROCEDIMIENTO
DECLARE
CURSOR CURSORMEMORIA;

SET AUTOPRINT ON;

VARIABLE CURSORMEMORIA SYS_REFCURSOR ;
EXECUTE PR_Mon_Imparte_clase(:CURSORMEMORIA);

*/

--RF-003
CREATE OR REPLACE PROCEDURE PR_Produc_vendidos
IS
BEGIN
/*SELECT sum(cantidad) FROM (SELECT Cantidad FROM productos P,
Cantidades C WHERE C.idproducto = P.idproducto);*/

SELECT nombre, sum(cantidad) FROM productos natural join
cantidades group by nombre;
END;
/

--RF-004
CREATE OR REPLACE PROCEDURE PR_Clientes_con_rut
IS
BEGIN
SELECT C.NOMBRE AS NOMBRE_C, R.NOMBRE AS NOMBRE_R FROM CLIENTES
C, RUTINAS R WHERE C.IDCLIENTE = R.IDCLIENTE;
END;
/

--RF-005
```

```
CREATE OR REPLACE PROCEDURE PR_For_cliente_rut (v_nombre IN
rutinas.nombre%TYPE, v_tipoDieta IN
rutinas.tipoDieta%TYPE,v_tipoEntrenamiento
IN rutinas.tipoEntrenamiento%TYPE, v_idPersonal IN
rutinas.idPersonal%TYPE, v_idCliente IN rutinas.idCliente%TYPE)
IS
BEGIN
INSERT INTO rutinas (nombre, tipoDieta, tipoEntrenamiento,
idPersonal, idCliente) VALUES (v_nombre, v_tipoDieta,
v_tipoEntrenamiento, v_idPersonal, v_idCliente);
COMMIT;
END;
/

--RF-006
CREATE OR REPLACE PROCEDURE PR_Clientes_con_rut
IS
BEGIN
SELECT NOMBRE, DIA, HORAINICIO, HORAFIN FROM CLASES C, HORARIOS
H WHERE C.IDCLASE = H.IDCLASE;
END;
/

--RN-003
CREATE OR REPLACE TRIGGER CantidadReservas
BEFORE INSERT ON Reservas
FOR EACH ROW
DECLARE
V_Res Integer;
BEGIN
 SELECT COUNT(*) INTO V_Res
 FROM Reservas R , Clases C
 WHERE R.idClase = C.idClase
 AND (dia, hora)
 OVERLAPS (:NEW.dia, :NEW.hora);
 IF (V_Res > participantes) THEN
 RAISE_APPLICATION_ERROR
 (-20600,'La clase ya está completa');
 END IF;
END;
/

--RN-004
CREATE OR REPLACE TRIGGER MaximoReservasPorCliente
BEFORE INSERT ON Reservas
FOR EACH ROW
DECLARE
V_Res Integer;
BEGIN
 SELECT COUNT(*) INTO V_Res
 FROM Reservas A
 WHERE idClase = :NEW.idClase
 AND (dia, hora)
 OVERLAPS (:NEW.dia, :NEW.hora);
 IF (V_Res >= 1) THEN
 RAISE_APPLICATION_ERROR
```

```
 (-20600,'Para dicho alojamiento ya existe una reserva
concurrente');
 END IF;
END;
/

--RN-006
CREATE OR REPLACE TRIGGER AvisoAdirector
BEFORE INSERT ON Cantidades
FOR EACH ROW
DECLARE
CANTIDAD_COMPRA INTEGER;
CANTIDAD_EN_STOCK INTEGER;
BEGIN
SELECT C.CANTIDAD AS CANTIDAD_COMPRADA, S.CANTIDAD AS
CANTIDAD_STOCK, NOMBRE, STOCKMINIMO FROM CANTIDADES C, PRODUCTOS
P, STOCK S WHERE P.IDPRODUCTO = C.IDPRODUCTO AND P.IDSTOCK =
S.IDSTOCK;
SELECT C.CANTIDAD  INTO CANTIDAD_COMPRA, S.CANTIDAD INTO
CANTIDAD_EN_STOCK FROM CANTIDADES C, PRODUCTOS P, STOCK S WHERE
P.IDPRODUCTO = :NEW.IDPRODUCTO AND P.IDSTOCK = S.IDSTOCK;
IF (CANTIDAD_EN_STOCK - CANTIDAD_C <= STOCKMINIMO) THEN
RAISE_APPLICATION_ERROR
(-20300,'Hay que avisar al director para realizar nuevo
pedido');
END IF;
UPDATE STOCK SET CANTIDAD_EN_STOCK = CANTIDAD_EN_STOCK -
CANTIDAD_COMPRA;
END;
/

--RN-008
CREATE OR REPLACE TRIGGER COMPROBAR_EXISTE_CLIENTE
BEFORE INSERT ON CLIENTES
FOR EACH ROW
DECLARE
V_DNI VARCHAR(9) := :NEW.DNI;
V_resultado VARCHAR(9);
BEGIN
SELECT DNI INTO V_resultado FROM CLIENTES WHERE DNI = V_DNI;
IF (V_resultado != NULL) THEN
RAISE_APPLICATION_ERROR
(-20300,'El cliente ya existe');
END IF;
END;
/
--RN-009
CREATE OR REPLACE TRIGGER INSCRIPCION_MONITOR
BEFORE INSERT ON CLASES
FOR EACH ROW
DECLARE
V_IDPERSONAL INTEGER;
V_DNI VARCHAR(9);
V_IDCARGO INTEGER;
V_DIA VARCHAR(10);
V_HORAINICIO VARCHAR(5);
```

```
BEGIN

SELECT IDPERSONAL INTO V_IDPERSONAL FROM PERSONAL WHERE
IDPERSONAL = V_IDPERSONAL;
SELECT DNI INTO v_DNI FROM PERSONAL WHERE DNI = v_DNI;
SELECT IDCARGO INTO v_IDCARGO FROM CARGOS WHERE IDCARGO =
v_IDCARGOS;
SELECT DIA INTO v_DIA FROM HORARIOS WHERE DIA = v_DIA;
SELECT HORAINICIO INTO v_HORAINICIO FROM HORARIOS WHERE
HORAINICIO = v_HORAINICIO;

SELECT P.IDPERSONAL AS IDPERSONAL_PERSONAL, DNI, IDCARGO, DIA,
HORAINICIO FROM CLASES C, PERSONAL P, HORARIOS H WHERE
P.IDPERSONAL = C.IDPERSONAL AND H.IDCLASE = C.IDCLASE;

IF (V_DIA = DIA AND V_HORAINICIO = HORAINICIO) THEN
RAISE_APPLICATION_ERROR
(-20300,'Este monitor ya tiene asignada una clase a dicha
hora');
END IF;
END;
/


--inserccion de valores en tabla
INSERT INTO clientes (dni, nombre, apellidos, correo, direccion,
telefono)
 VALUES ('12345678X', 'Juan', 'Blanco', 'blanco@us.es', 'Nueva
25', '(34)622334455');
INSERT INTO clientes (dni, nombre, apellidos, correo, direccion,
telefono)
 VALUES ('54673436X', 'Raul', 'Garcia', 'garcia@us.es',
'Andalucía 35', '(34)632452119');
 INSERT INTO clientes (dni, nombre, apellidos, correo,
direccion, telefono)
 VALUES ('76983436D', 'Juan', 'Jiminez', 'jimenez@us.es',
'sevilla 35', '(34)632452222');
INSERT INTO ofertas (nombre, precio)
 VALUES ('Activación', '20');
INSERT INTO ofertas (nombre, precio)
 VALUES ('Musculación', '20');
INSERT INTO matriculas (fechaInicio, fechaFin, tipoPago,
IdCliente, IdOferta)
 VALUES (TO_DATE('2018-12-20', 'YYYY-MM-DD'), TO_DATE('2019-01-
20', 'YYYY-MM-DD'), 'tarjeta', 1, 1);
INSERT INTO matriculas (fechaInicio, fechaFin, tipoPago,
IdCliente, IdOferta)
 VALUES (TO_DATE('2018-12-17', 'YYYY-MM-DD'), TO_DATE('2019-01-
17', 'YYYY-MM-DD'), 'metalico', 2, 2);
INSERT INTO aulas (nombre)
 VALUES ('aula 1');
 INSERT INTO aulas (nombre)
 VALUES ('aula 2');
INSERT INTO cargos (nombre, salario)
 VALUES ('Monitor', '550');
INSERT INTO cargos (nombre, salario)
```

```
 VALUES ('Recepcionista', '700');
 INSERT INTO cargos (nombre, salario)
 VALUES ('Director', '1000');
INSERT INTO departamentos (nombre)
 VALUES ('Deportivo');
INSERT INTO departamentos (nombre)
 VALUES ('Limpieza');
INSERT INTO proveedores (nombre, correo, direccion, telefono)
 VALUES ('ProductoMusculacion', 'progym@gmail.com', 'Cruz 23',
'(34)643452119');
INSERT INTO proveedores (nombre, correo, direccion, telefono)
 VALUES ('ProductoPerdida', 'producgym@gmail.com', 'Mujer 23',
'(34)609052119');
INSERT INTO personal (dni, nombre, apellidos, telefono,
direccion, correo, IdDepartamento, IdCargo)
 VALUES ('42312465F', 'Maria', 'Ramirez', '(34)622376545',
'Pineda 15', 'ramirez@gmail.es', 002, 001);
INSERT INTO personal (dni, nombre, apellidos, telefono,
direccion, correo, IdDepartamento, IdCargo)
 VALUES ('44442465M', 'Antonio', 'Gutierrez', '(34)733376545',
'Palmera 7', 'gutierrez@gmail.es', 001, 001);
 INSERT INTO personal (dni, nombre, apellidos, telefono,
direccion, correo, IdDepartamento, IdCargo)
 VALUES ('94442465H', 'Jesus', 'Sanchez', '(34)787376545',
'Manzano 7', 'Sanchez@gmail.es', 001, 003);
INSERT INTO clases (nombre, participantes, duracion, IdPersonal,
IdAula)
 VALUES ('Zumba', 20 , 'hora', 1, 2);
INSERT INTO clases (nombre, participantes, duracion, IdPersonal,
IdAula)
 VALUES ('Aerobic', 18, 'hora', 2, 1);
INSERT INTO horarios (dia, horaInicio, horaFin, Idclase)
 VALUES ('Martes', '17:00', '18:00', 2);
INSERT INTO horarios (dia, horaInicio, horaFin, Idclase)
 VALUES ('Lunes', '19:00', '20:00', 1);
INSERT INTO reservas (hora, dia, Idclase, Idcliente)
 VALUES ('19:00', TO_DATE('2019-01-18', 'YYYY-MM-DD'), 2, 1);
INSERT INTO reservas (hora, dia, Idclase, Idcliente)
 VALUES ('17:00', TO_DATE('2019-01-19', 'YYYY-MM-DD'), 2, 2);
INSERT INTO rutinas (nombre, tipoDieta, tipoEntrenamiento,
IDpersonal, IDcliente)
 VALUES ('Gan_musc', 'Hipercalorica', 'Hipertrofia', 1, 1);
INSERT INTO rutinas (nombre, tipoDieta, tipoEntrenamiento,
IDpersonal, IDcliente)
 VALUES ('Per_Peso', 'Hipocalorica', 'Definicion', 2, 2);
INSERT INTO contratos ( fechaInicio, fechaFin, IDpersonal,
IDcargo)
 VALUES (TO_DATE('2018-12-20', 'YYYY-MM-DD'), TO_DATE('2019-01-
20', 'YYYY-MM-DD'), 1, 1);
INSERT INTO contratos ( fechaInicio, fechaFin, IDpersonal,
IDcargo)
 VALUES (TO_DATE('2019-01-01', 'YYYY-MM-DD'), TO_DATE('2019-03-
30', 'YYYY-MM-DD'), 2, 2);
INSERT INTO stock (cantidad, IDpersonal)
 VALUES ('25', 2);
INSERT INTO stock (cantidad, IDpersonal)
```

```
 VALUES ('15', 2);
INSERT INTO productos (nombre, precio, stockMinimo, IDstock,
IDproveedor)
 VALUES ('Barrita energetica', 3, 15, 2, 1);
INSERT INTO productos (nombre, precio, stockMinimo, IDstock,
IDproveedor)
 VALUES ('Agua', 1, 25, 1, 2);
INSERT INTO compras (fecha, Idpersonal, Idcliente)
 VALUES (TO_DATE('2019-01-01', 'YYYY-MM-DD'), 2, 1);
INSERT INTO compras (fecha, Idpersonal, Idcliente)
 VALUES (TO_DATE('2019-01-11', 'YYYY-MM-DD'), 2, 2);
INSERT INTO cantidades (cantidad, idProducto, idCompra)
 VALUES (2, 2, 1);
INSERT INTO cantidades (cantidad, IdProducto, IdCompra)
 VALUES (3, 1, 1);
INSERT INTO pagos (cantidadPago, fechaPago, IDpersonal1,
IDpersonal2)
 VALUES ('550', TO_DATE('2019-01-01', 'YYYY-MM-DD'), 1, 3);
INSERT INTO pagos (cantidadPago, fechaPago, IDpersonal1,
IDpersonal2)
 VALUES ('700', TO_DATE('2019-02-01', 'YYYY-MM-DD'), 2, 3);


--SCRIPTT DE PRUEBA(CLIENTES)

CREATE OR REPLACE
FUNCTION ASSERT_EQUALS(SALIDA BOOLEAN, SALIDAESPERADA BOOLEAN)
RETURN VARCHAR2 AS
BEGIN
IF(SALIDA = SALIDAESPERADA) THEN
RETURN 'EXITO';
ELSE
RETURN 'FALLO';
END IF;
END ASSERT_EQUALS;


CREATE OR REPLACE PACKAGE PRUEBAS_CLIENTES AS
PROCEDURE INICIALIZAR;
PROCEDURE INSERTAR
(NOMBRE_PRUEBA VARCHAR2, v_dni IN clientes.dni%TYPE, v_nombre IN
clientes.nombre%TYPE,v_apellidos IN clientes.apellidos%TYPE,
v_correo IN clientes.correo%TYPE, v_direccion IN
clientes.direccion%TYPE, v_telefono IN clientes.telefono%TYPE,
SALIDAESPERADA BOOLEAN);
PROCEDURE ACTUALIZAR
(NOMBRE_PRUEBA VARCHAR2, V_IDCLIENTE INTEGER,  v_nombre IN
clientes.nombre%TYPE, SALIDAESPERADA BOOLEAN);
PROCEDURE ELIMINAR
(NOMBRE_PRUEBA VARCHAR2, V_IDCLIENTE INTEGER, SALIDAESPERADA
BOOLEAN);
END PRUEBAS_CLIENTES;

CREATE OR REPLACE PACKAGE BODY PRUEBAS_CLIENTES AS
PROCEDURE INICIALIZAR AS
BEGIN
```

```
DELETE FROM CLIENTES;
END INICIALIZAR;
PROCEDURE INSERTAR (NOMBRE_PRUEBA VARCHAR2, v_dni IN
clientes.dni%TYPE, v_nombre IN clientes.nombre%TYPE,v_apellidos
IN clientes.apellidos%TYPE,
v_correo IN clientes.correo%TYPE, v_direccion IN
clientes.direccion%TYPE, v_telefono IN clientes.telefono%TYPE,
SALIDAESPERADA BOOLEAN) AS
SALIDA BOOLEAN := TRUE;
CLIENTE CLIENTES%ROWTYPE;
V_IDCLIENTE INTEGER;
BEGIN
INSERT INTO CLIENTES VALUES (SEC_CLI.NEXTVAL, v_dni, v_nombre,
v_apellidos, v_correo, v_direccion, v_telefono);
V_IDCLIENTE := SEC_CLI.CURRVAL;
SELECT * INTO CLIENTE FROM CLIENTES WHERE IDCLIENTE =
V_IDCLIENTE;
IF (CLIENTE.NOMBRE<>V_NOMBRE) THEN
SALIDA := FALSE;
END IF;
COMMIT WORK;
DBMS_OUTPUT.PUT_LINE(NOMBRE_PRUEBA || ':' ||
ASSERT_EQUALS(SALIDA,SALIDAESPERADA));
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(NOMBRE_PRUEBA || ':' ||
ASSERT_EQUALS(FALSE,SALIDAESPERADA));
ROLLBACK;
END INSERTAR;


PROCEDURE ACTUALIZAR (NOMBRE_PRUEBA VARCHAR2, V_IDCLIENTE
INTEGER,  v_nombre IN clientes.nombre%TYPE, SALIDAESPERADA
BOOLEAN) AS
SALIDA BOOLEAN := TRUE;
CLIENTE CLIENTES%ROWTYPE;
BEGIN
UPDATE CLIENTES SET NOMBRE = V_NOMBRE WHERE IDCLIENTE =
V_IDCLIENTE;
SELECT * INTO CLIENTE FROM CLIENTES WHERE IDCLIENTE =
V_IDCLIENTE;
IF (CLIENTE.NOMBRE<>V_NOMBRE) THEN
SALIDA := FALSE;
END IF;
COMMIT WORK;
DBMS_OUTPUT.PUT_LINE(NOMBRE_PRUEBA || ':' ||
ASSERT_EQUALS(SALIDA,SALIDAESPERADA));
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(NOMBRE_PRUEBA || ':' ||
ASSERT_EQUALS(FALSE,SALIDAESPERADA));
ROLLBACK;
END ACTUALIZAR;

PROCEDURE ELIMINAR (NOMBRE_PRUEBA VARCHAR2, V_IDCLIENTE INTEGER,
SALIDAESPERADA BOOLEAN) AS
```

```
SALIDA BOOLEAN := TRUE;
N_CLIENTES INTEGER;
BEGIN
DELETE FROM CLIENTES WHERE IDCLIENTE = V_IDCLIENTE;
SELECT COUNT(*) INTO N_CLIENTES FROM CLIENTES WHERE IDCLIENTE =
V_IDCLIENTE;
IF (N_CLIENTES<>0) THEN
SALIDA := FALSE;
END IF;
COMMIT WORK;
DBMS_OUTPUT.PUT_LINE(NOMBRE_PRUEBA || ':' ||
ASSERT_EQUALS(SALIDA,SALIDAESPERADA));
EXCEPTION
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(NOMBRE_PRUEBA || ':' ||
ASSERT_EQUALS(FALSE,SALIDAESPERADA));
ROLLBACK;
END ELIMINAR;
END;
/

/* FICHERO PRINCIPAL PRUEBAS*/
/* ACTIVAR SALIDA DE TEXTO POR PANTALLA*/
SET SERVEROUTPUT ON;

DECLARE
    id_cliente integer;
    id_cli_paco integer;
    id_cli_manuel integer;

BEGIN
/* PRUEBAS DE LAS OPERACIONES SOBRE LA TABLA CLIENTES*/
    PRUEBAS_CLIENTES.INICIALIZAR;
    PRUEBAS_CLIENTES.INSERTAR ('PRUEBA1- INSERCION
CLIENTE','78345678C', 'Paco', 'Gallardo', 'gallardo@us.es',
'Nueva 25', '(34)622334457', true);
    id_cliente := SEC_CLI.CURRVAL;
   PRUEBAS_CLIENTES.INSERTAR ('PRUEBA2- INSERCION CLIENTE NOMBRE
NULL','72245678M', 'Pepito', NULL, 'pepito@us.es', 'Nueva 25',
'(34)611334457', false);
     PRUEBAS_CLIENTES.ACTUALIZAR ('PRUEBA3- ACTUALIZACION NOMBRE
CLIENTE', id_cliente, 'Andres', true);
    PRUEBAS_CLIENTES.ACTUALIZAR ('PRUEBA4- ACTUALIZACION NOMBRE
CLIENTE null', id_cliente, null, false);
    PRUEBAS_CLIENTES.ELIMINAR ('PRUEBA5- ELIMINAR CLIENTE',
id_cliente,true);
END;


-- CREACIÓN DE CURSORES Y CONSULTAS

DECLARE
    CURSOR c IS
    SELECT dni, nombre FROM clientes ORDER BY nombre;
    BEGIN
```

```
    DBMS_OUTPUT.PUT_LINE('Los 3 primeros clientes por orden
alfabético');
    FOR fila IN c LOOP
    EXIT WHEN c%ROWCOUNT > 3;
    DBMS_OUTPUT.PUT_LINE(fila.dni||' '||fila.nombre);
    END LOOP;
END;
```