Universidad Autónoma de Yucatán

Proyecto Integrador

Diseño de Bases de Datos

Profesor: Dr. Raúl Antonio Aguilar Vera

Integrantes:

Manuel Martin Rico

Salma Rubi Wabi

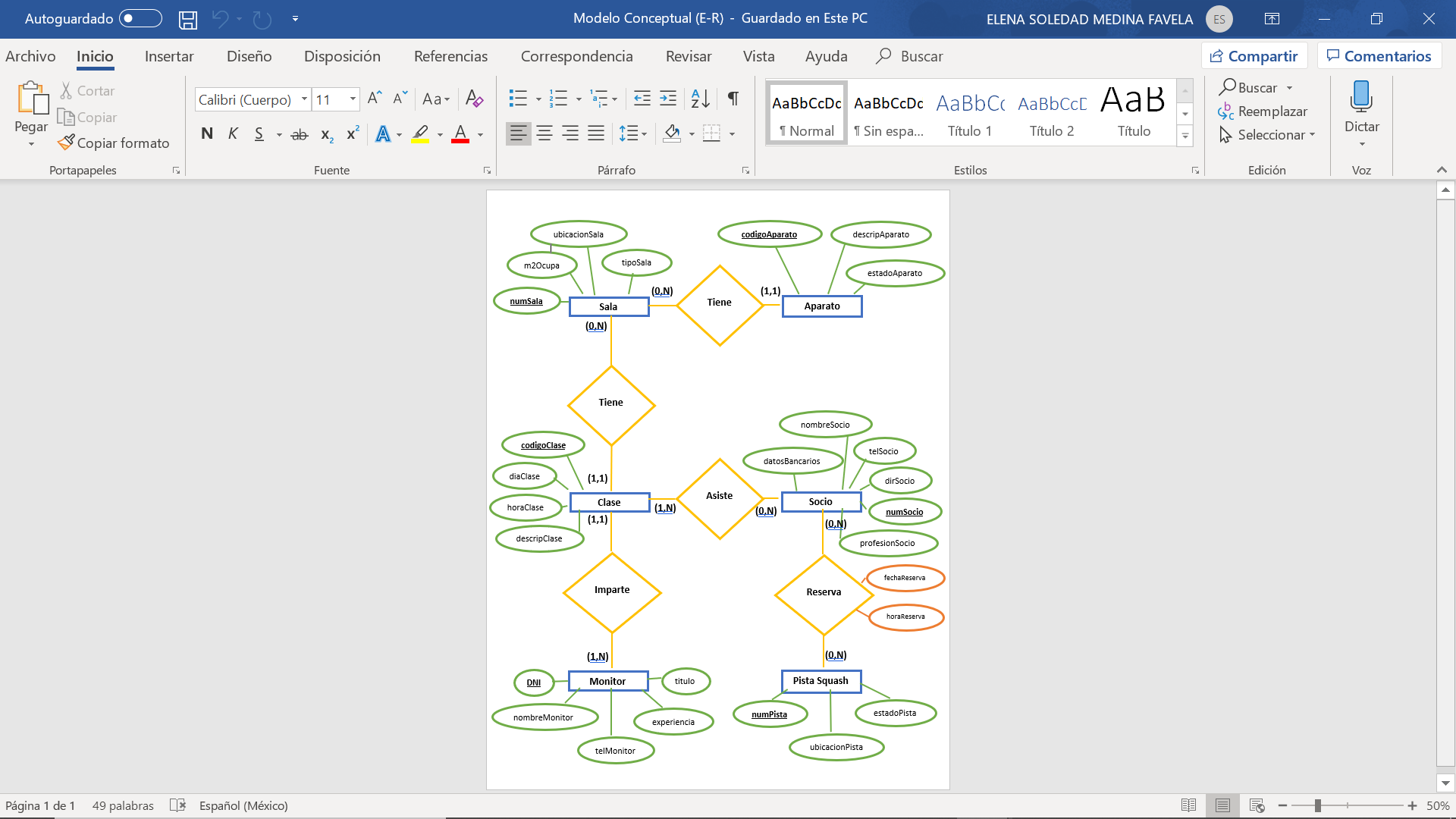
Zeus Jhonatan Sarmiento Serrano

Elena Soledad Medina Favela

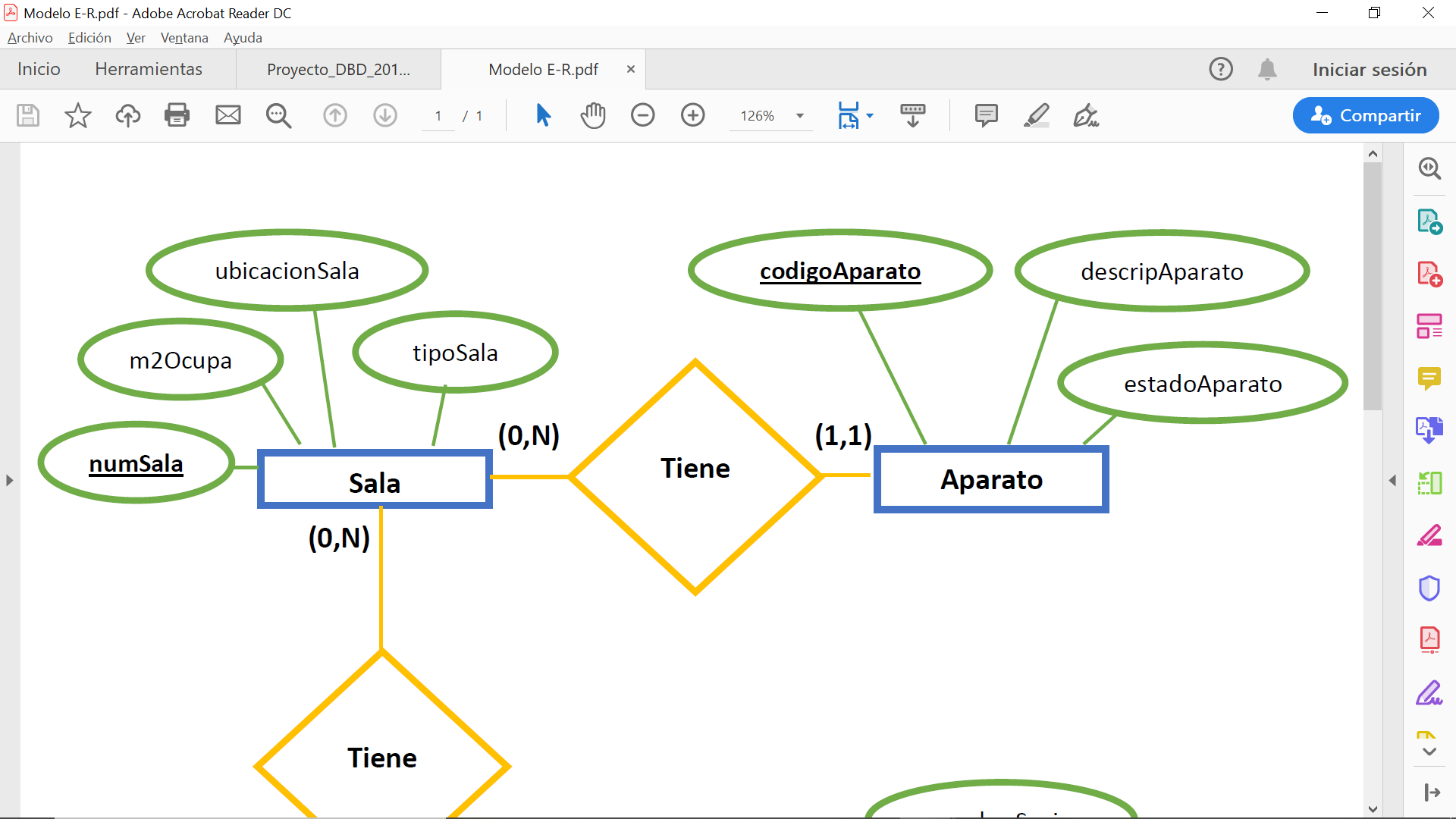
Jorge Armando Garcia Bellos

Fecha: 5 de diciembre de 2019

I. Elabore EL Diseño Lógico de la BD para el Sistema de Información del gimnasio “Zagros Sport” utilizando el procedimiento estudiado en la Unidad 3 (Modelo Conceptual con E/R y Reglas de Transformación para obtener el Modelo Lógico).

**Modelo Conceptual**

**Modelo Lógico**

1. Transformación de la relación:

Aplicando R1 (Representación tabular de los conjuntos de entidades fuertes), obtenemos la(s) siguiente(s) tabla(s):

|  |
| --- |
| **Sala** |
| numSala | M2Ocupa | ubicacionSala | tipoSala |

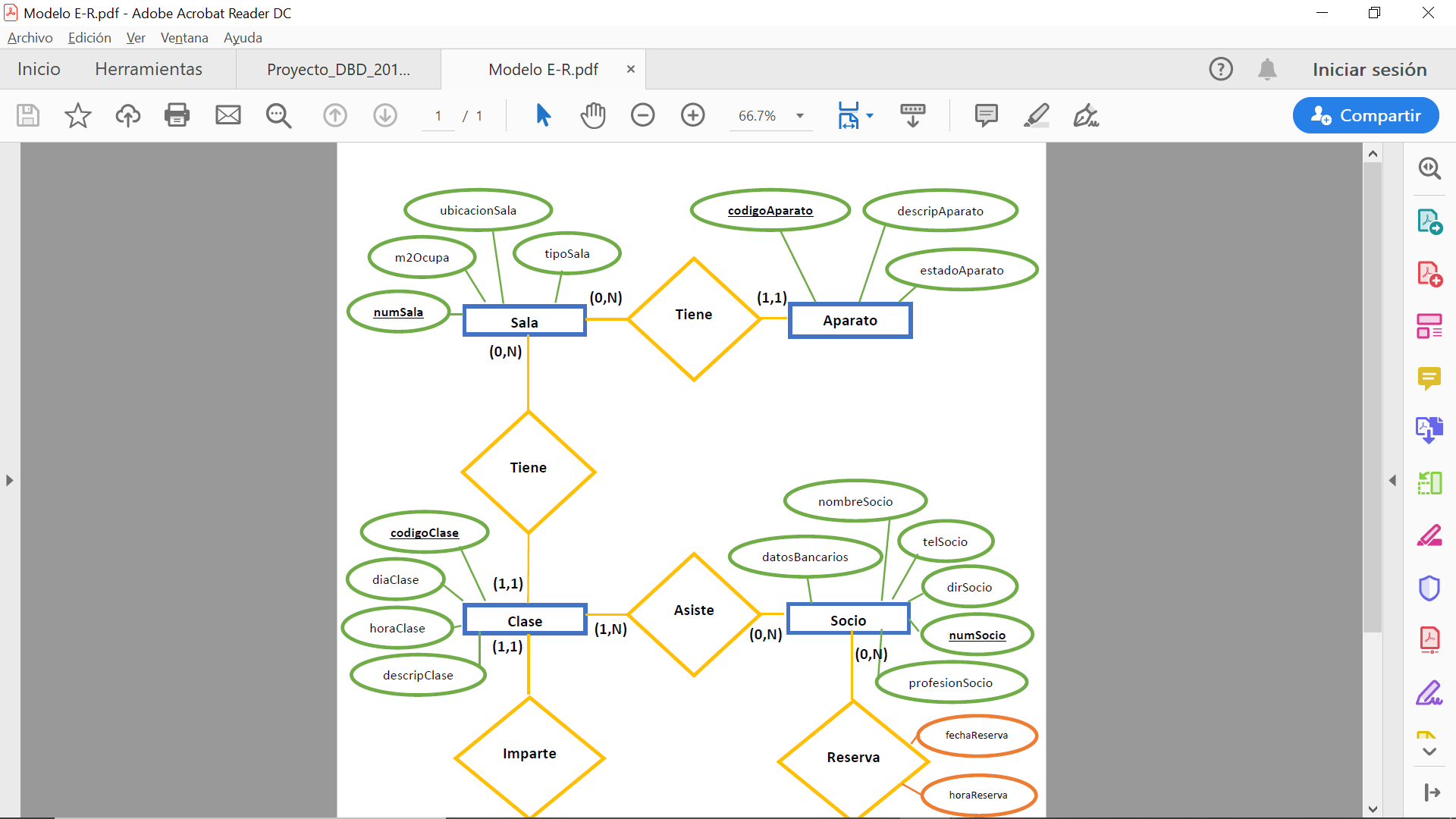
|  |
| --- |
| **Aparato** |
| codigoAparato | descripAparato | estadoAparato |

Aplicando R2 (Representación tabular de los conjuntos de relaciones, para interrelaciones n-1, 1-n), obtenemos la(s) siguiente(s) tabla(s):

|  |
| --- |
| **Sala** |
| numSala | M2Ocupa | ubicacionSala | tipoSala |

|  |
| --- |
| **Aparato** |
| codigoAparato | descripAparato | estadoAparato | numSala |

1. Transformación de la relación:



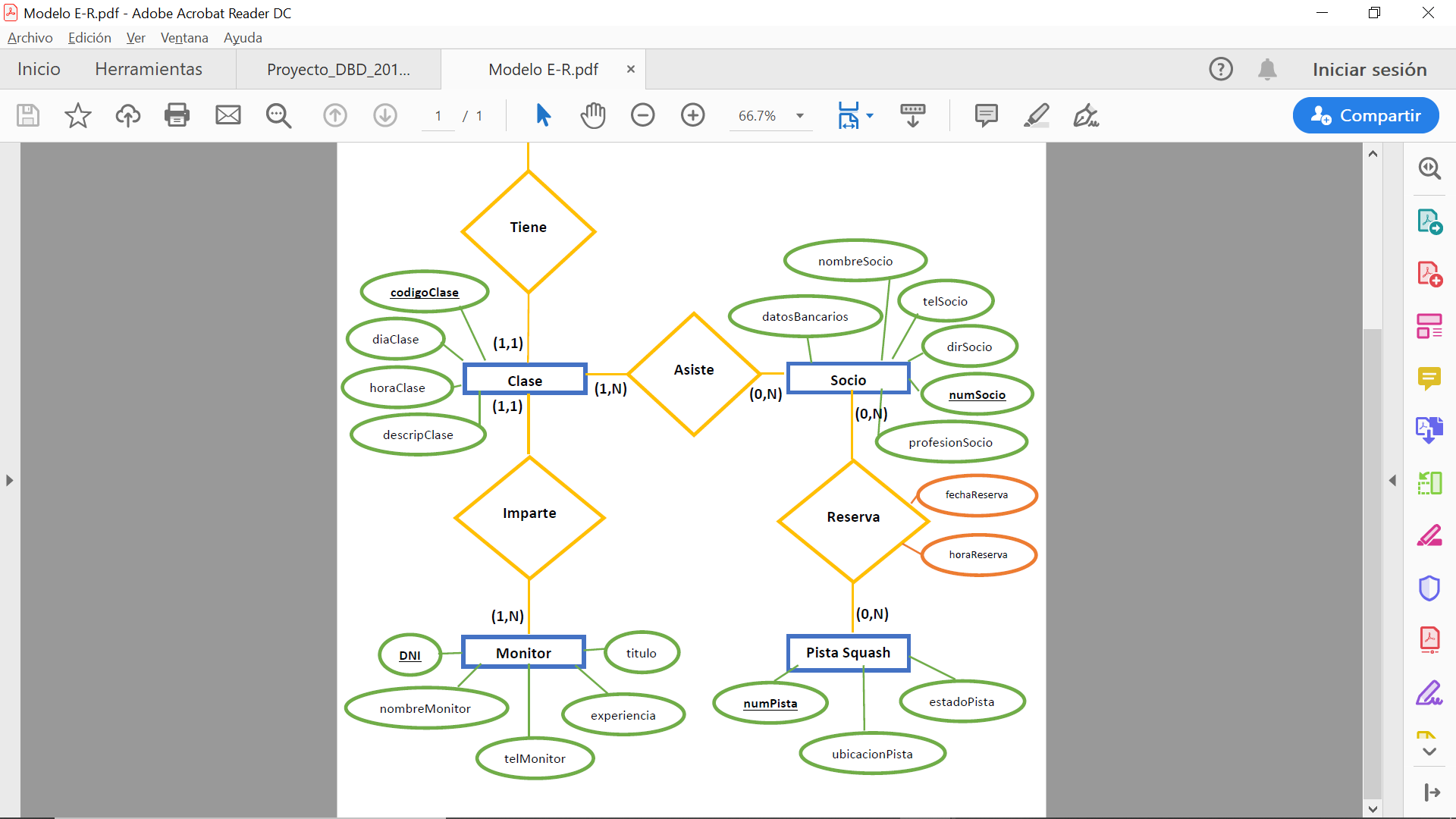
Aplicando R1 (Representación tabular de los conjuntos de entidades fuertes), obtenemos la(s) siguiente(s) tabla(s):

|  |
| --- |
| **Clase** |
| codigoClase | diaClase | horaClase | descripClase |

Aplicando R2 (Representación tabular de los conjuntos de relaciones, para interrelaciones n-1, 1-n), obtenemos la(s) siguiente(s) tabla(s):

|  |
| --- |
| **Clase** |
| codigoClase | diaClase | horaClase | descripClase | numSala |

1. Transformación de la relación:



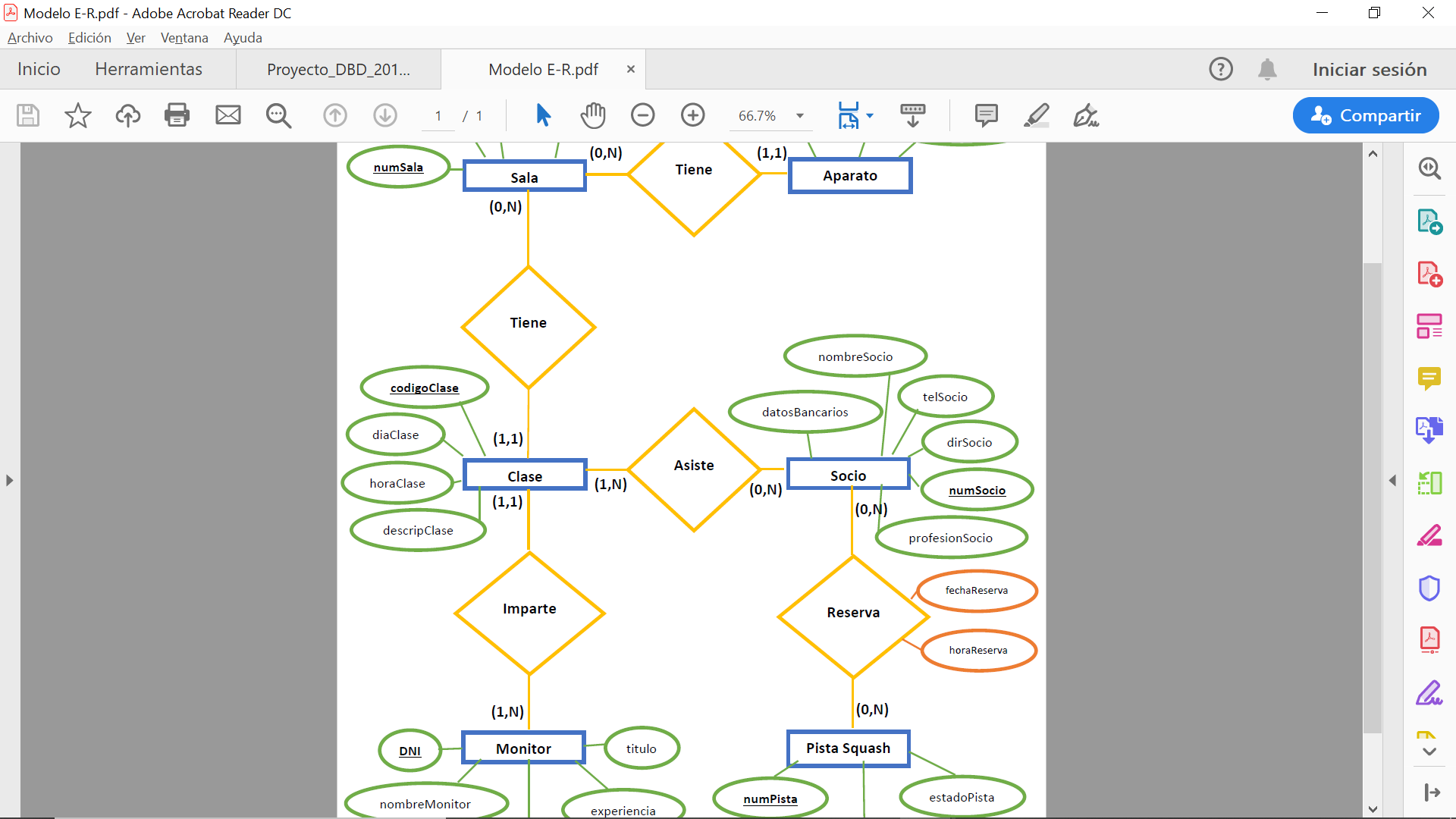
Aplicando R1 (Representación tabular de los conjuntos de entidades fuertes), obtenemos la(s) siguiente(s) tabla(s):

|  |
| --- |
| **Monitor** |
| DNI | nombreMonitor | telMonitor | titulo | experiencia |

Aplicando R2 (Representación tabular de los conjuntos de relaciones, para interrelaciones n-1, 1-n), obtenemos la(s) siguiente(s) tabla(s):

|  |
| --- |
| **Clase** |
| codigoClase | diaClase | horaClase | descripClase | numSala | DNIMonitor |

1. Transformación de la relación:



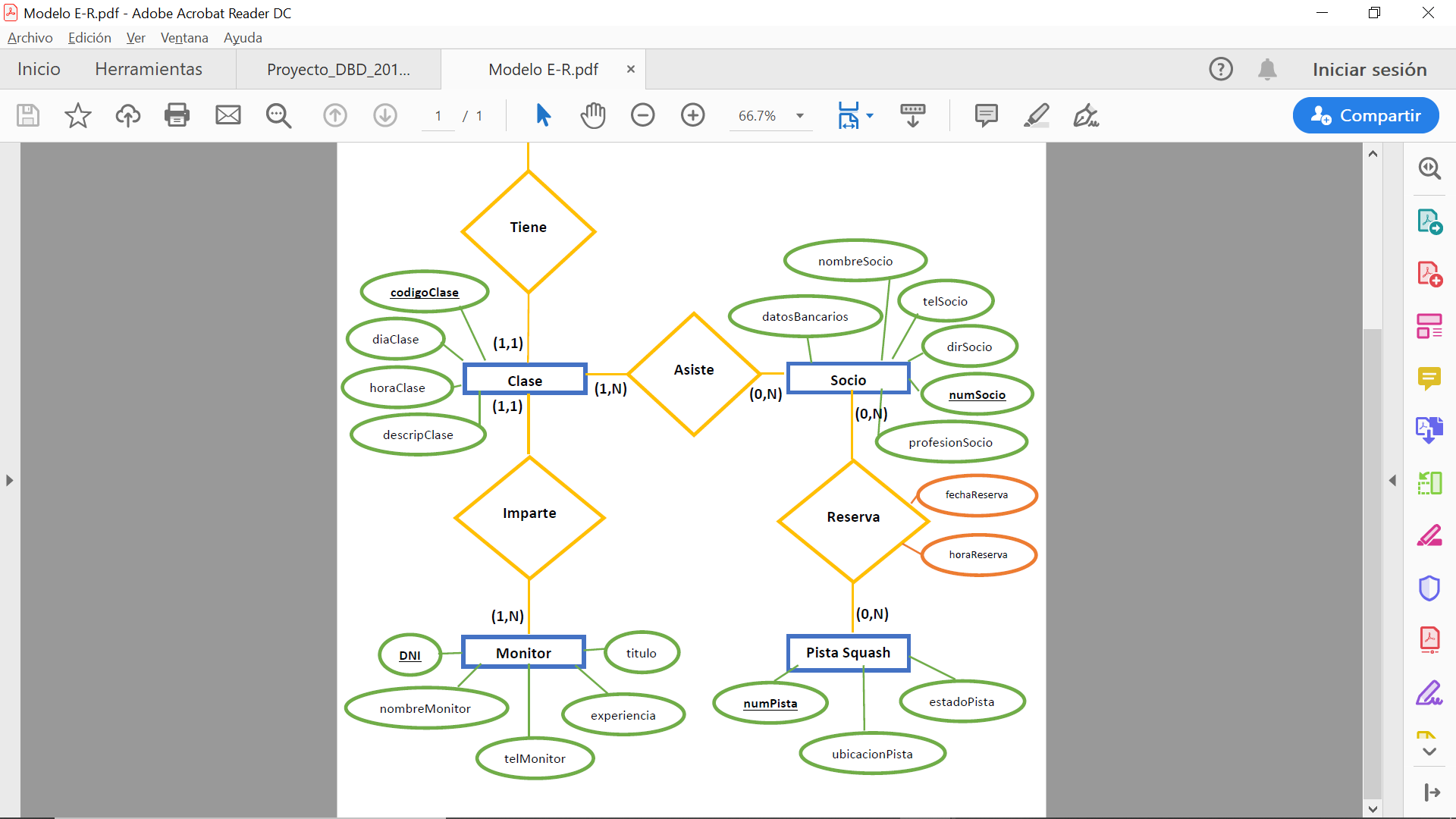
Aplicando R1 (Representación tabular de los conjuntos de entidades fuertes), obtenemos la(s) siguiente(s) tabla(s):

|  |
| --- |
| **Socio** |
| numSocio | nombreSocio | telSocio | dirSocio | datosBancarios | profesionSocio |

Aplicando R2 (Representación tabular de los conjuntos de relaciones, para interrelaciones n-n), obtenemos la(s) siguiente(s) tabla(s):

|  |
| --- |
| **Asiste** |
| codigoClase | numSocio |

1. Transformación de la relación:



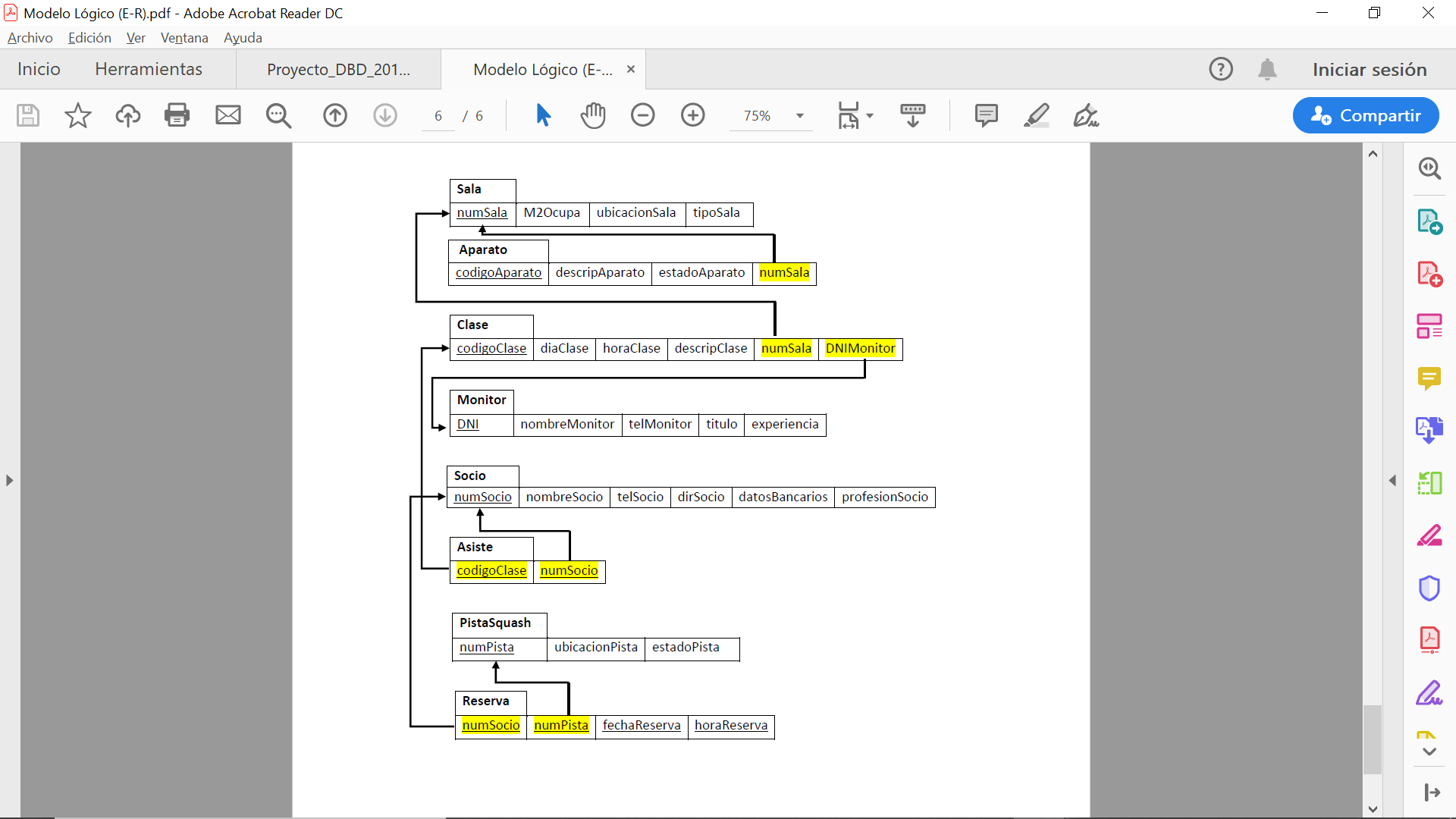
Aplicando R1 (Representación tabular de los conjuntos de entidades fuertes), obtenemos la(s) siguiente(s) tabla(s):

|  |
| --- |
| **PistaSquash** |
| numPista | ubicacionPista | estadoPista |

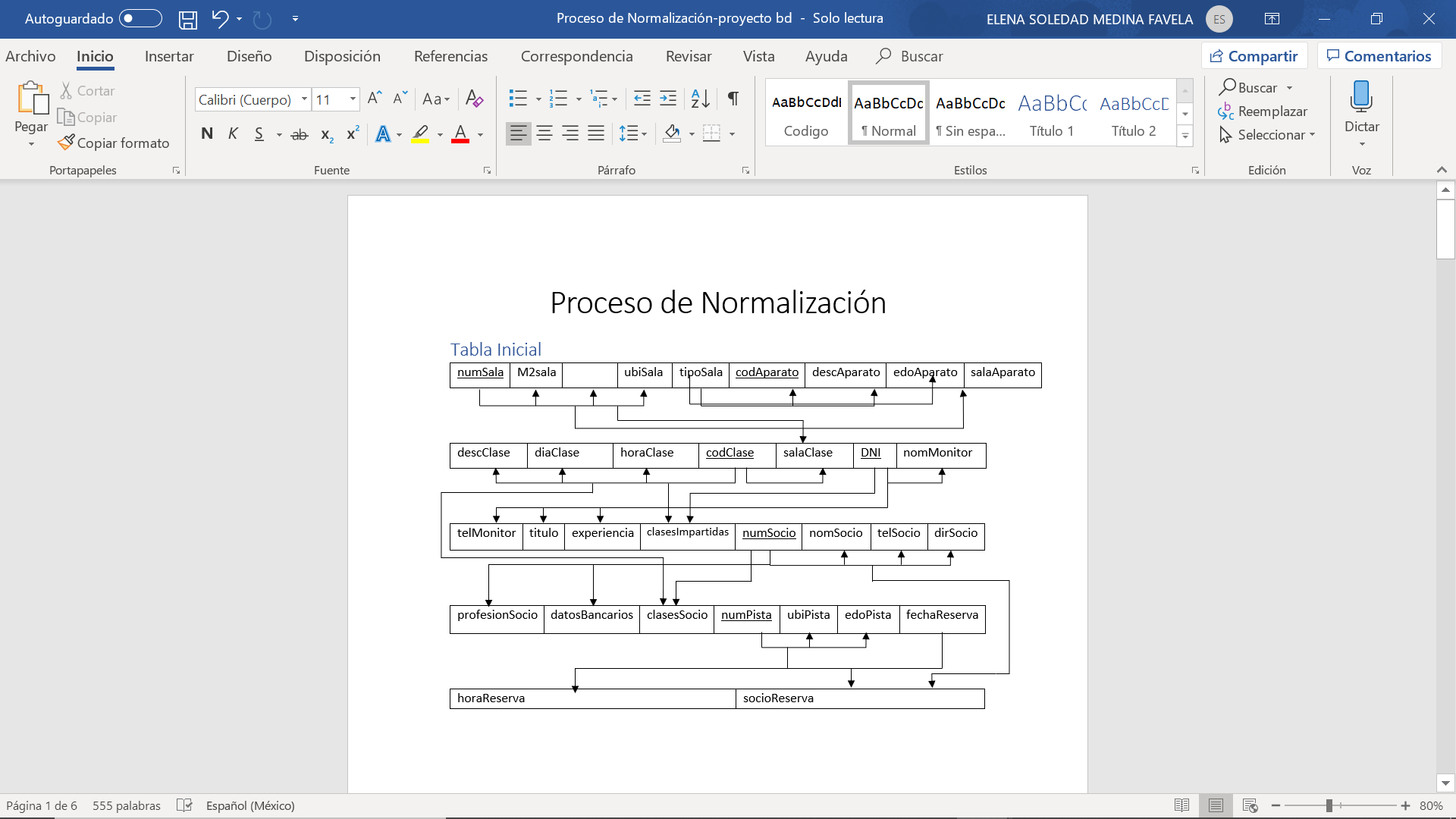
Aplicando R2 (Representación tabular de los conjuntos de relaciones, para interrelaciones n-n), obtenemos la(s) siguiente(s) tabla(s):

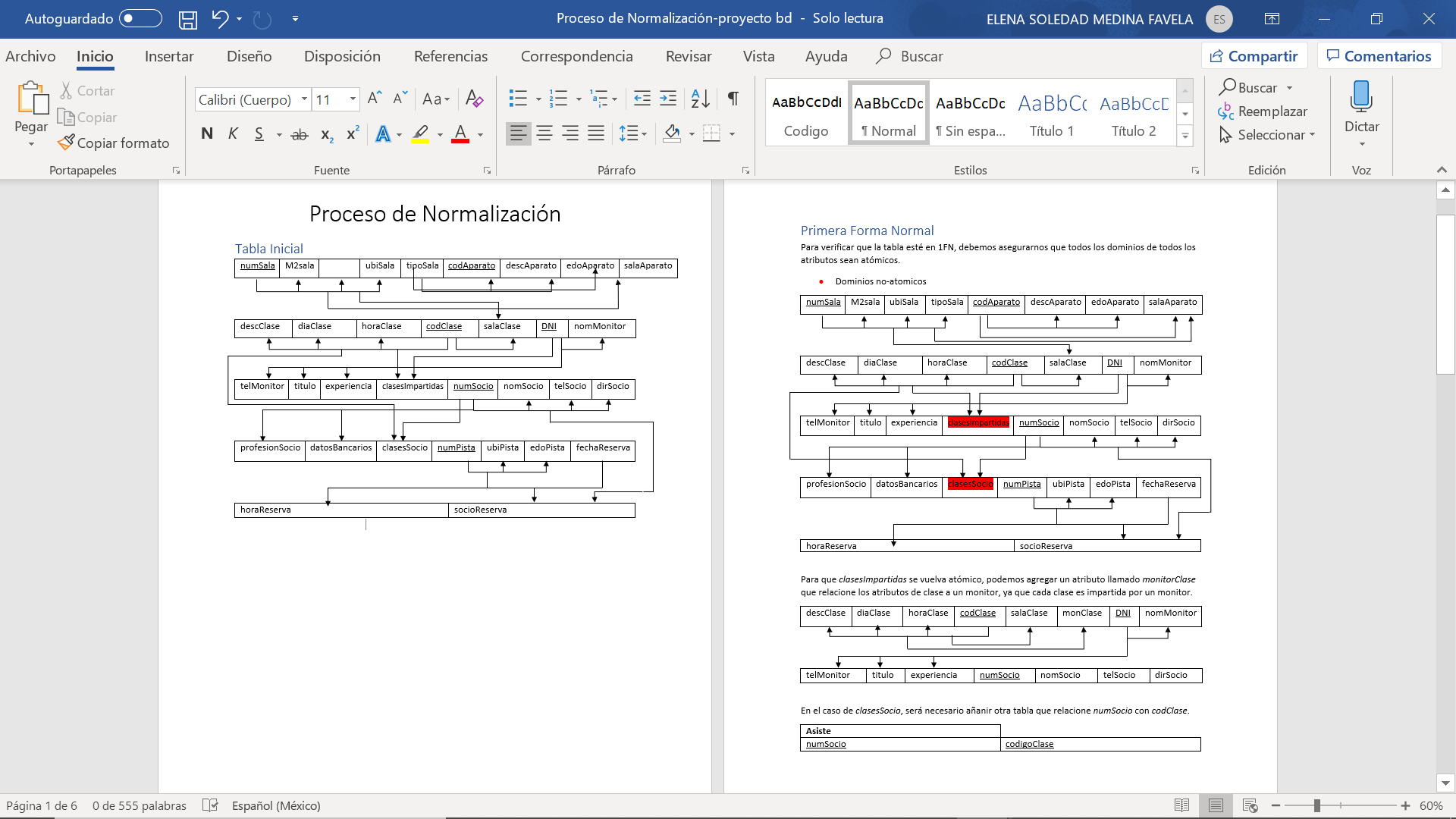
|  |
| --- |
| **Reserva** |
| numSocio | numPista | fechaReserva | horaReserva |

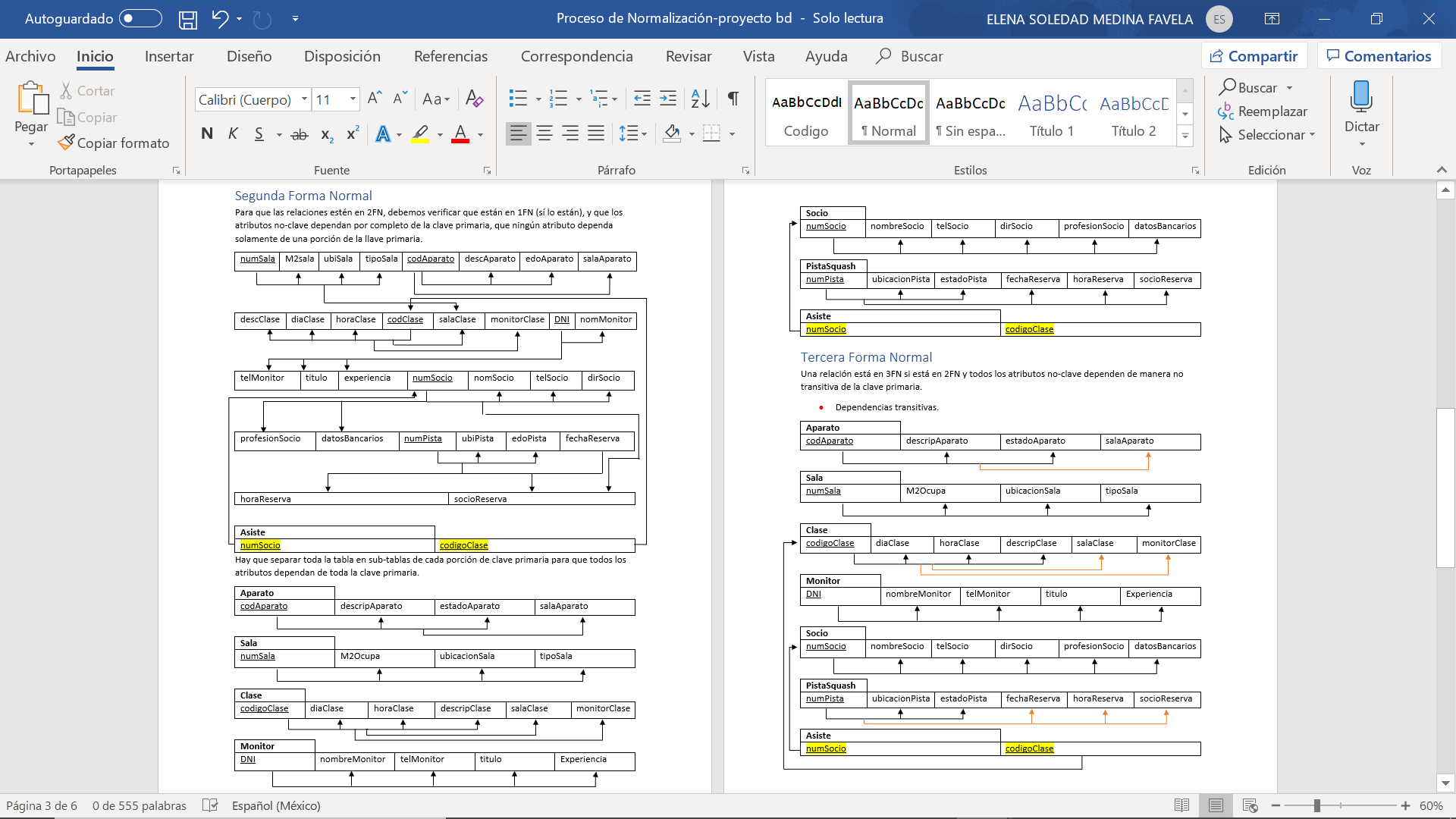
Después de concluir el proceso de transformación, el modelo lógico es el siguiente:

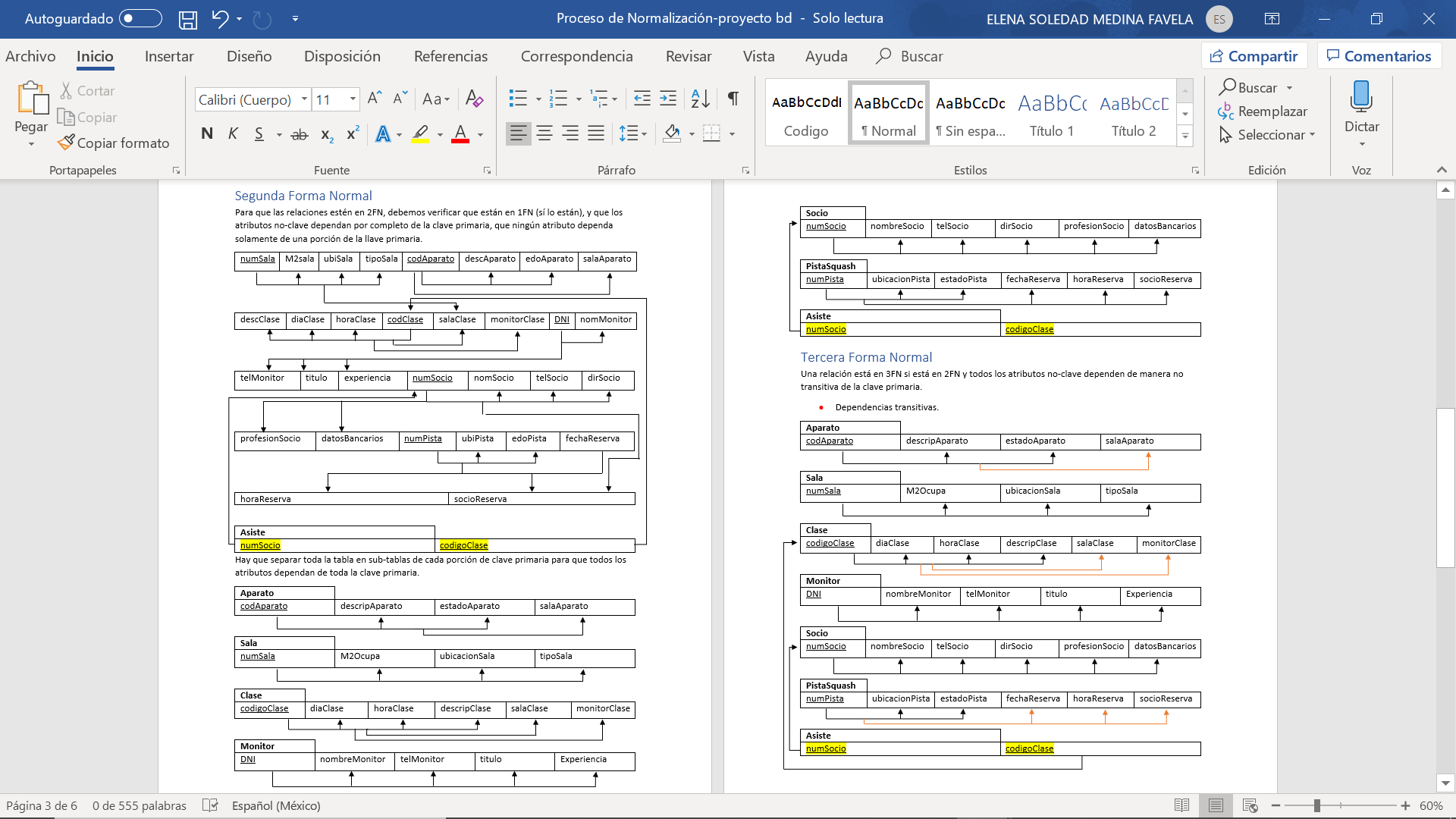


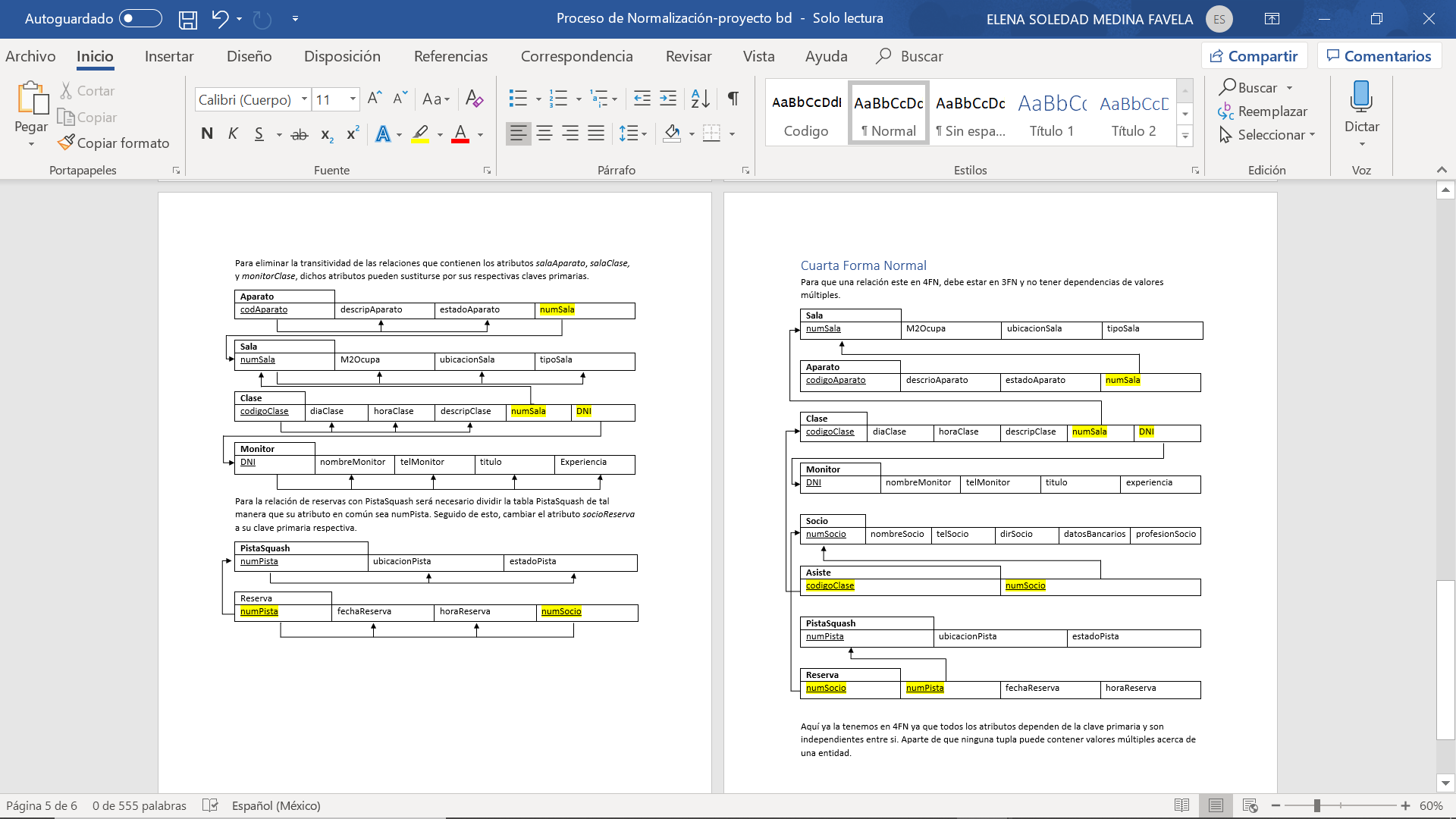
II. Elabore EL Diseño Lógico de la BD para el Sistema de Información del gimnasio “*Zagros Sport*” utilizando el procedimiento estudiado en la Unidad 4 (Aplicar el Proceso de Normalización para obtener el Modelo lógico con relaciones validadas hasta la 4FN).

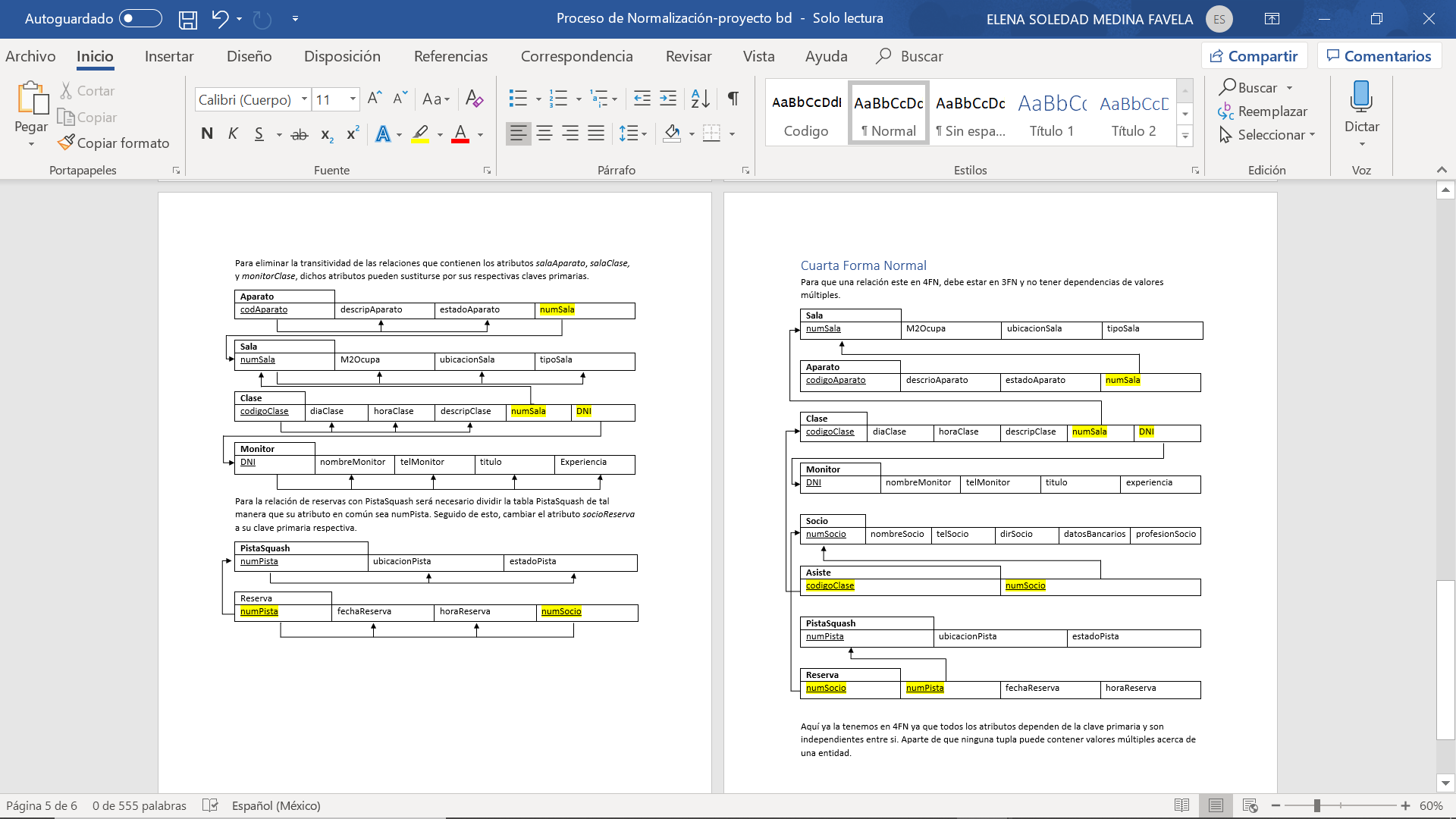












III. Reflexione con sus compañeros de equipo y elabore un reporte en torno a las bondades e implicaciones de cada uno de los dos procesos de diseño estudiados. Con base en lo adquirido durante el curso y con la presente reflexión, sugiera (con argumentos) el procedimiento a utilizar para el Diseño de una BDR.

Para el proceso de diseño de la base de datos del proyecto integrador implementamos dos métodos: la normalización y el diseño conceptual entidad-relación con una posterior transformación al modelo relacional. A continuación, hemos recopilado las primeras opiniones originales de cada uno de los integrantes al respecto del proceso:

* **Salma Wabi:** En mi opinión es mejor el proceso de normalización, ya que es menos ambiguo y para llevar una relación a su forma normal se debe satisfacer cierto conjunto de condiciones preestablecidas que sirven para eliminar la redundancia de los datos. Al hacer un modelo ER es más difícil evitar esto y es más fácil cometer errores en el diseño.
* **Jorge García:** La normalización me pareció un mejor método, ya que tiene una estructura y unos pasos más claros y el resultado final también me parece más completo, Aparte de que en mi opinión es más fácil de aplicar. Por su parte, el modelo ER me pareció muy abstracto, se puede tener varios modelos que apliquen correctamente para el problema a resolver.
* **Elena Medina:** En mi opinión cada forma la usaría en diferentes escenarios. El modelo ER lo usaría para proyectos pequeños, ya que siento que es muy fácil y rápido de implementar y la parte gráfica ayuda mucho a entender el modelo rápidamente. En cambio, la normalización la utilizaría en proyectos más complejos, ya que, al tener un proceso predefinido es más sencillo manejar la información y los resultados son más precisos que en el modelo ER.
* **Zeus Sarmiento:** Por mi parte, elijo el modelo entidad - relación, principalmente por su sencillez; también empleado un proceso sistemático que resulta fácil de entender, los pasos relacionados directamente pero aun así se puede trabajar de manera independiente en cada uno de ellos.
* **Manuel Martin:** El proceso de normalización se me hizo más preciso, si bien no siento que sea tan sencillo como el modelo entidad – relación, pienso que este nos permite llegar a un resultado más preciso y óptimo. Pienso que el punto fuerte de la normalización es el hecho de contar con un proceso algorítmico riguroso y teóricamente firme, en cambio, el modelo ER depende mucho de la percepción que pueda llegar a tener el diseñador de la base de datos.

Después de discutirlo entre todo el equipo, hemos llegado a la siguiente conclusión:

El proceso puede depender mucho del desarrollador, ya que como se vio anteriormente, a algunos se les facilita más un método que a otros.

En cuanto a los procesos de diseño:

**Modelo Conceptual ER y transformación.**

Ventajas:

* Es más libre a la creatividad del diseñador, permite expresar bien las ideas que se tiene del sistema.
* El mapeo final de los elementos es comprensible y le da una idea general sencilla a cualquier elemento del equipo de trabajo.
* Las relaciones y las entidades (bien hechas) pueden explicar por si mismo la funcionalidad del sistema e incluso pueden explicar la idea principal de la unidad dependiente que hará uso de esta.

Desventajas:

* En un sistema muy grande en donde no se tenga una buena división de componentes, es más fácil que el diseño se salga de control y se termine con un sistema poco escalable y para nada entendible.
* Puede llegar a ser muy abstracto para algunos, esto principalmente al momento de su creación y diseño.
* Se puede llegar a varios diseños que se adapten bien el problema, si bien esto puede ser considerado como una ventaja, para el equipo representa un margen de inseguridad entre cual de todos es el óptimo.

**Procedimiento de normalización.**

Ventajas:

* Es más riguroso, se tiene la certeza de que se llega al resultado óptimo.
* Se tiene un proceso algoritmo claro.
* No es tan abstracto, ya que se manejan los datos crudos que nos conciernen.

Desventajas:

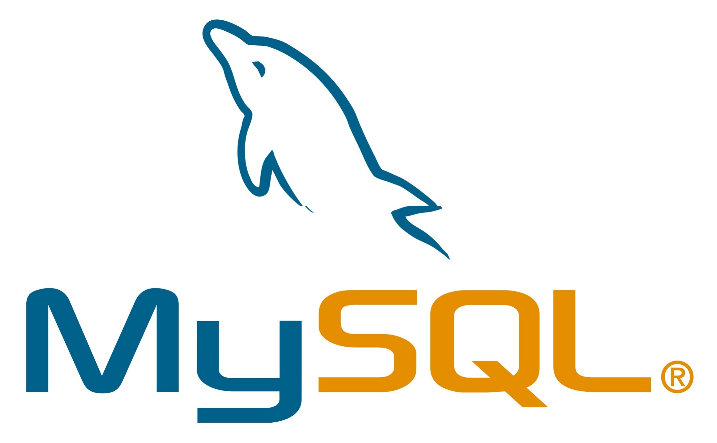
* El inicio del proceso puede ser complicado, la unión de los datos y los pasos principales pueden no ser tan sencillos.
* No resulta eficiente en problemas pequeños, como una base de datos de usuarios y comentarios genérica.
* No ofrece tantas libertades como el proceso anterior.

Como reflexión final, el equipo piensa que ambos métodos pueden ser buenas herramientas para el proceso de diseño de bases de datos, en específico pensamos que el modelo relacional puede servirnos para sistemas pequeños, donde necesitemos una representación gráfica y se tenga cierto margen para el diseñador para expresar la idea que se tiene de la solución. El proceso de normalización puede ser el mejor cuando se tienen distintos componentes codependientes en un sistema y se necesita la solución más precisa, aparte de buscar rigurosidad en los procesos.

Llegando a un mutuo acuerdo, el equipo ha decido utilizar el proceso de normalización para este proyecto, ya que requerimos tener el resultado más concreto posible y evitar las ambigüedades.

IV. Realice una investigación documental sobre 3 SMBD relacionales, y elabore un reporte indicando: datos de autoría, características y tabla comparativa. Seleccione (con argumentos) uno de dichos SMBD.

**MySQL**

MySQL es un sistema de gestión de base de datos relacionales de código abierto. Es uno de los pocos gestores que se ejecutan en prácticamente todas las plataformas, desde Linux, hasta Windows. Está considerada como la base de datos de código abierto más popular del mundo.

* Modelo primario: relacional
* Desarrollador: Oracle
* Fecha de primer lanzamiento: 1995
* Licencia: Open source
* Lenguajes implementados: C y C++

**Características**

Fue desarrollada inicialmente por MySQL AB, luego adquirida por Sun Microsystems y luego posteriormente por Oracle Coporation en 2010. Es utilizado para sitios web de gran tamaño, algunos ejemplos pueden ser: youtube, Wikipedia, Facebook.

Es mayormente utilizado para en conjunción con servidores web aplicado a aplicaciones o CMS para sitios online. Se le relaciona mucho con PHP.

¿Por qué mysql es tan popular para sitios web? En el desarrollo web no se suele presentar un número elevado de modificaciones de datos, pero si se presenta la lectura frecuente de estos; MySQL presenta la BD ideal para esto.

Entre otras características se encuentran:

* Múltiples motores de almacenamiento para cada tabla.
* Ejecución de transacciones y uso de llaves foráneas.
* Amplio subconjunto del lenguaje SQL.
* Disponible para casi todas las plataformas y sistemas.
* Búsqueda e indexación de campos de texto.
* Tablas hash en memorias temporales.
* Se implementa B-Tree para búsquedas más rápidas.

MySQL es la opción principal en cuanto al panorama didáctico y recreativo, pero en muchos ámbitos empresariales ha quedo en desuso. Entre otras desventajas podemos encontrar las típicas que presentan los sistemas open source: falta de documentación y falta de un grupo de mantenimiento, prácticamente la comunidad de desarrollo depende de si misma para esto.

**PostgreSQL**

PostgreSQL es un sistema gestor de bases de datos relacional orientada a objetos, se dice orientada a objetos ya que incluye características de este paradigma, como la herencia, las funciones, etc.

* Modelo primario: relacional.
* Desarrollador: PostgreSQL Global Development Group
* Fecha de primer lanzamiento: 1989
* Licencia: Open source
* Lenguajes implementados: C

**Características**

Es gratuito y libre, además de que hoy nos ofrece una gran cantidad de opciones avanzadas. Se le considera el motor de base de datos (open source) más avanzado de la actualidad.

Al igual que MySQL, tiene soporte oficial en muchas plataformas: Windows, Ubuntu, Solarios, etc.

Como se mencionó antes, una de las ventajas de PostgreSQL es que aproxima los datos a un modelo orientado a objetos, esto hace que sea capaz de manejar rutinas más complejas. Como parte del paradigma, permita utilizar la herencia para que una tabla (padre) pueda compartir datos a otras tablas (hijos), las tuplas insertadas en las tablas hijas será insertadas en la tabla padre respectivamente.

Entre otras características se encuentra:

* Es altamente extensible.
* Contiene tecnología MVCC (Control de concurrencia multi-version)
* Mejor documentada que MySQL. Comunidades de desarrollo en distintos idiomas.

Al igual que con MySQL, PostgreSQL puede tener las mismas fallas que llega a tener cualquier sistema open source, aunque este mejor documentado y tenga más apoyo entre la comunidad de desarrolladores. Otras desventajas:

* Consume muchos recursos.
* Es de 2 a 3 veces más lento que MySQL.

**Microsoft SQL Server**

SQL Server es un sistema de gestión de base de datos relacional (RDBMS)producido por Microsoft. Su principal lenguaje de consulta es Transact-SQL, una aplicación de las normas ANSI / ISO estándar Structured Query Language (SQL)utilizado por ambas Microsoft y Sybase.

* Modelo primario: relacional
* Desarrollador: Microsoft
* Fecha de primer lanzamiento: 1989
* Licencia: comercial
* Lenguajes implementados: C++

**Características**

* Soporte de transacciones
* Escalabilidad, estabilidad y seguridad
* Soporta procedimientos almacenados
* Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente
* Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información
* Además, permite administrar información de otros servidores de datos

Entre las desventajas de SQL server, se encuentra el hecho de que no es Open source, se necesita una licencia para utilizar la versión empresarial. La documentación, si bien existe, la curva de aprendizaje puede ser muy pronunciada en comparación.

**Tabla comparativa**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Microsoft SQL Server** | **MySQL** | **PostgreSQL** |
| Modelo de datos | Si | Si | Si |
| Lenguajes soportados | C#  C++  Delphi  Go  Java  JavaScript (Node.js)  JavaScript  Python  PHP  R  Ruby  Visual Basic | Ada  C  C#  C++  D  Delphi  Eiffel  Erlang  Haskell  Java  JavaScript (Node.js)  Objective-C  OCaml  Perl  PHP  Python  Ruby  Scheme  Tcl | .Net  C  C++  Delphi  Java  JavaScript (Node.js)  Perl  PHP  Python  Tcl |
| Versión actual | SQL Server 2019 | 8.0.18 | 12.1 |
| Licencia | Comercial | Open Source | Open Source |
| Basado en la nube | No | No | No |
| Sistema operativo con soporte | Linux  Windows | FreeBSD  Linux  OS X  Solaris  Windows | FreeBSD  HP-UX  Linux  NetBSD  OpenBSD  OS X  Solaris  Unix  Windows |
| Modelo de base de datos secundario | Orientado a documentos  Orientado a gráficos | Orientado a documentos | Orientado a documentos |

El equipo ha decido utilizar MySQL como sistema gestor ¿Por qué?

1. MySQL es el sistema gestor por excelencia para fines didácticos.
2. El equipo ya tiene formación con este.
3. Se busca mantener la filosofía Open Source.
4. Sería exagerado utilizar tanto PostgreSQL, como SQL Server; estos no se adaptan correctamente a las necesidades del proyecto. MySQL es la herramienta exacta que se busca.

Profundizando un poco más en el punto 4, SQL server es un gestor con fines empresariales, para grandes volúmenes de datos, incluso podría manejar Big Data. En el caso de que el modelo relacional, posteriormente utilizado, nos hayamos encontrado con características del paradigma orientado a objetos, utilizaríamos PostgreSQL, pero no es el caso. La simplicidad de MySQL va de la mano con la simplicidad del diseño y del problema a resolver. Como conclusión, se elige MySQL únicamente por que el equipo ha decidido que es suficiente para poder llevar a cabo el proyecto.

V. Utilizando el Sistema Manejador de Bases de Datos seleccionado, Implemente el Diseño Físico de la Base de Datos del problema descrito (incluya las sentencias en el reporte).

-- phpMyAdmin SQL Dump

-- version 4.9.1

-- https://www.phpmyadmin.net/

--

-- Servidor: 127.0.0.1

-- Tiempo de generación: 04-12-2019 a las 08:47:43

-- Versión del servidor: 10.4.8-MariaDB

-- Versión de PHP: 7.3.10

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Base de datos: `zagrossport`

--

CREATE DATABASE IF NOT EXISTS `zagrossport` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci;

USE `zagrossport`;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `aparato`

--

CREATE TABLE `aparato` (

`codigoAparato` varchar(20) NOT NULL,

`descripAparato` varchar(50) NOT NULL,

`estadoAparato` varchar(20) NOT NULL,

`numSala` int(5) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `asiste`

--

CREATE TABLE `asiste` (

`codigoClase` varchar(20) NOT NULL,

`numSocio` int(6) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `clase`

--

CREATE TABLE `clase` (

`codigoClase` varchar(20) NOT NULL,

`diaClase` varchar(12) NOT NULL,

`horaClase` varchar(12) NOT NULL,

`descripClase` varchar(50) NOT NULL,

`numSala` int(5) NOT NULL,

`DNIMonitor` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `monitor`

--

CREATE TABLE `monitor` (

`DNI` varchar(20) NOT NULL,

`nombreMonitor` varchar(30) NOT NULL,

`telMonitor` varchar(12) NOT NULL,

`titulo` varchar(50) NOT NULL,

`experiencia` varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `pistasquash`

--

CREATE TABLE `pistasquash` (

`numPista` int(5) NOT NULL,

`ubicacionPista` varchar(20) NOT NULL,

`estadoPista` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `reserva`

--

CREATE TABLE `reserva` (

`numSocio` int(6) NOT NULL,

`numPista` int(5) NOT NULL,

`fechaReserva` date NOT NULL,

`horaReserva` varchar(10) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `sala`

--

CREATE TABLE `sala` (

`numSala` int(5) NOT NULL,

`M2Ocupa` int(6) NOT NULL,

`ubicacionSala` varchar(50) NOT NULL,

`tipoSala` varchar(20) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `socio`

--

CREATE TABLE `socio` (

`numSocio` int(6) NOT NULL,

`nombreSocio` varchar(30) NOT NULL,

`telSocio` varchar(12) NOT NULL,

`dirSocio` varchar(50) NOT NULL,

`datosBancarios` varchar(50) NOT NULL,

`profesionSocio` varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--

-- Índices para tablas volcadas

--

--

-- Indices de la tabla `aparato`

--

ALTER TABLE `aparato`

ADD PRIMARY KEY (`codigoAparato`),

ADD KEY `numSala` (`numSala`);

--

-- Indices de la tabla `asiste`

--

ALTER TABLE `asiste`

ADD PRIMARY KEY (`codigoClase`,`numSocio`),

ADD KEY `codigoClase` (`codigoClase`),

ADD KEY `numSocio` (`numSocio`);

--

-- Indices de la tabla `clase`

--

ALTER TABLE `clase`

ADD PRIMARY KEY (`codigoClase`),

ADD KEY `numSala` (`numSala`),

ADD KEY `DNIMonitor` (`DNIMonitor`);

--

-- Indices de la tabla `monitor`

--

ALTER TABLE `monitor`

ADD PRIMARY KEY (`DNI`);

--

-- Indices de la tabla `pistasquash`

--

ALTER TABLE `pistasquash`

ADD PRIMARY KEY (`numPista`);

--

-- Indices de la tabla `reserva`

--

ALTER TABLE `reserva`

ADD PRIMARY KEY (`numSocio`,`numPista`,`fechaReserva`,`horaReserva`),

ADD KEY `numSocio` (`numSocio`),

ADD KEY `numPista` (`numPista`);

--

-- Indices de la tabla `sala`

--

ALTER TABLE `sala`

ADD PRIMARY KEY (`numSala`);

--

-- Indices de la tabla `socio`

--

ALTER TABLE `socio`

ADD PRIMARY KEY (`numSocio`);

--

-- Restricciones para tablas volcadas

--

--

-- Filtros para la tabla `aparato`

--

ALTER TABLE `aparato`

ADD CONSTRAINT `aparato\_ibfk\_1` FOREIGN KEY (`numSala`) REFERENCES `sala` (`numSala`);

--

-- Filtros para la tabla `asiste`

--

ALTER TABLE `asiste`

ADD CONSTRAINT `asiste\_ibfk\_1` FOREIGN KEY (`codigoClase`) REFERENCES `clase` (`codigoClase`),

ADD CONSTRAINT `asiste\_ibfk\_2` FOREIGN KEY (`numSocio`) REFERENCES `socio` (`numSocio`);

--

-- Filtros para la tabla `clase`

--

ALTER TABLE `clase`

ADD CONSTRAINT `clase\_ibfk\_1` FOREIGN KEY (`numSala`) REFERENCES `sala` (`numSala`),

ADD CONSTRAINT `clase\_ibfk\_2` FOREIGN KEY (`DNIMonitor`) REFERENCES `monitor` (`DNI`);

--

-- Filtros para la tabla `reserva`

--

ALTER TABLE `reserva`

ADD CONSTRAINT `reserva\_ibfk\_1` FOREIGN KEY (`numSocio`) REFERENCES `socio` (`numSocio`),

ADD CONSTRAINT `reserva\_ibfk\_2` FOREIGN KEY (`numPista`) REFERENCES `pistasquash` (`numPista`);

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

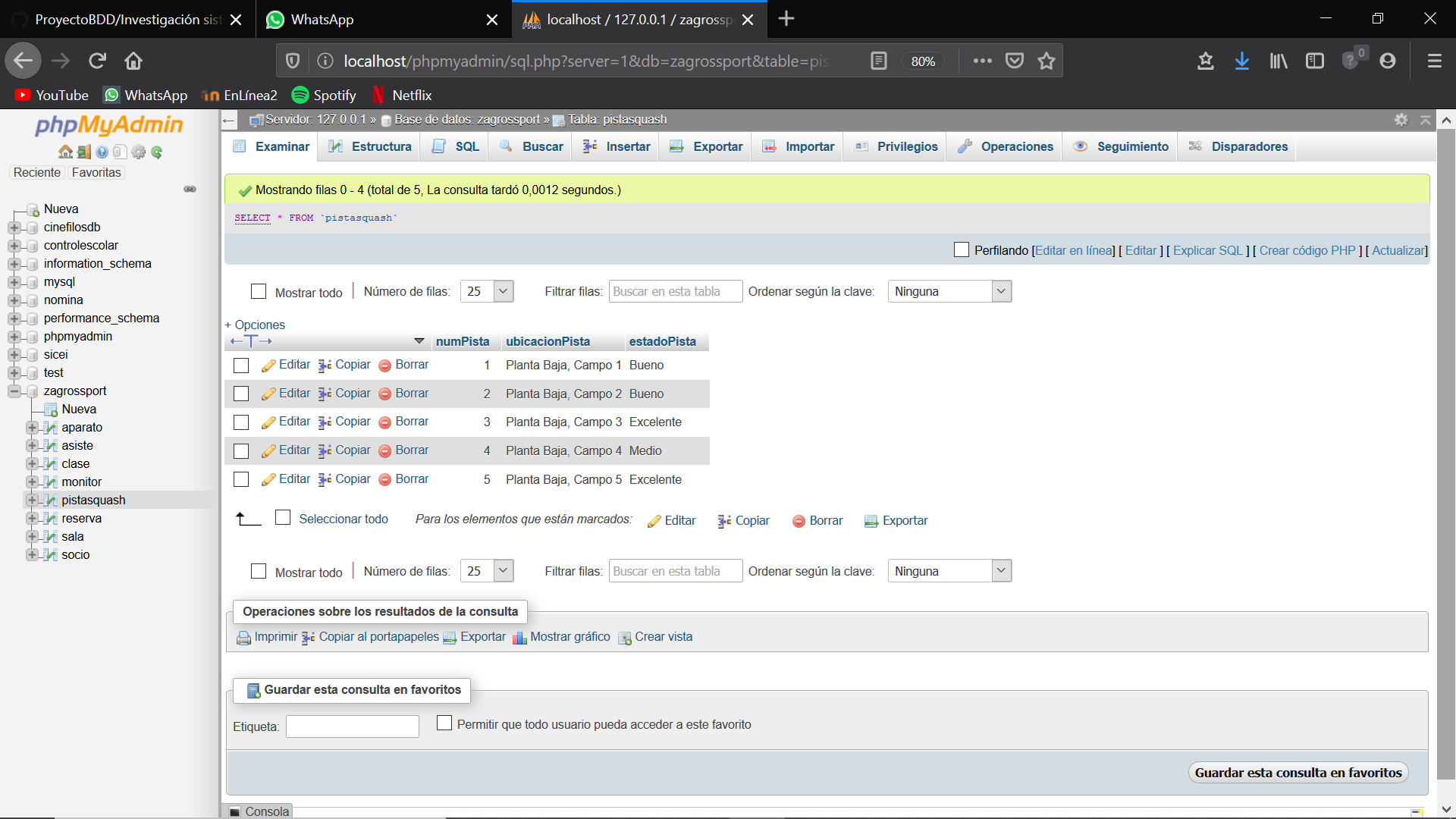
/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;



VI. Utilizando el Sistema Manejador de Bases de Datos seleccionado, capture un conjunto de tuplas relacionadas con la información que dicha BD mantiene (incluya un par de ejemplos de las sentencias utilizadas para la captura en el reporte).

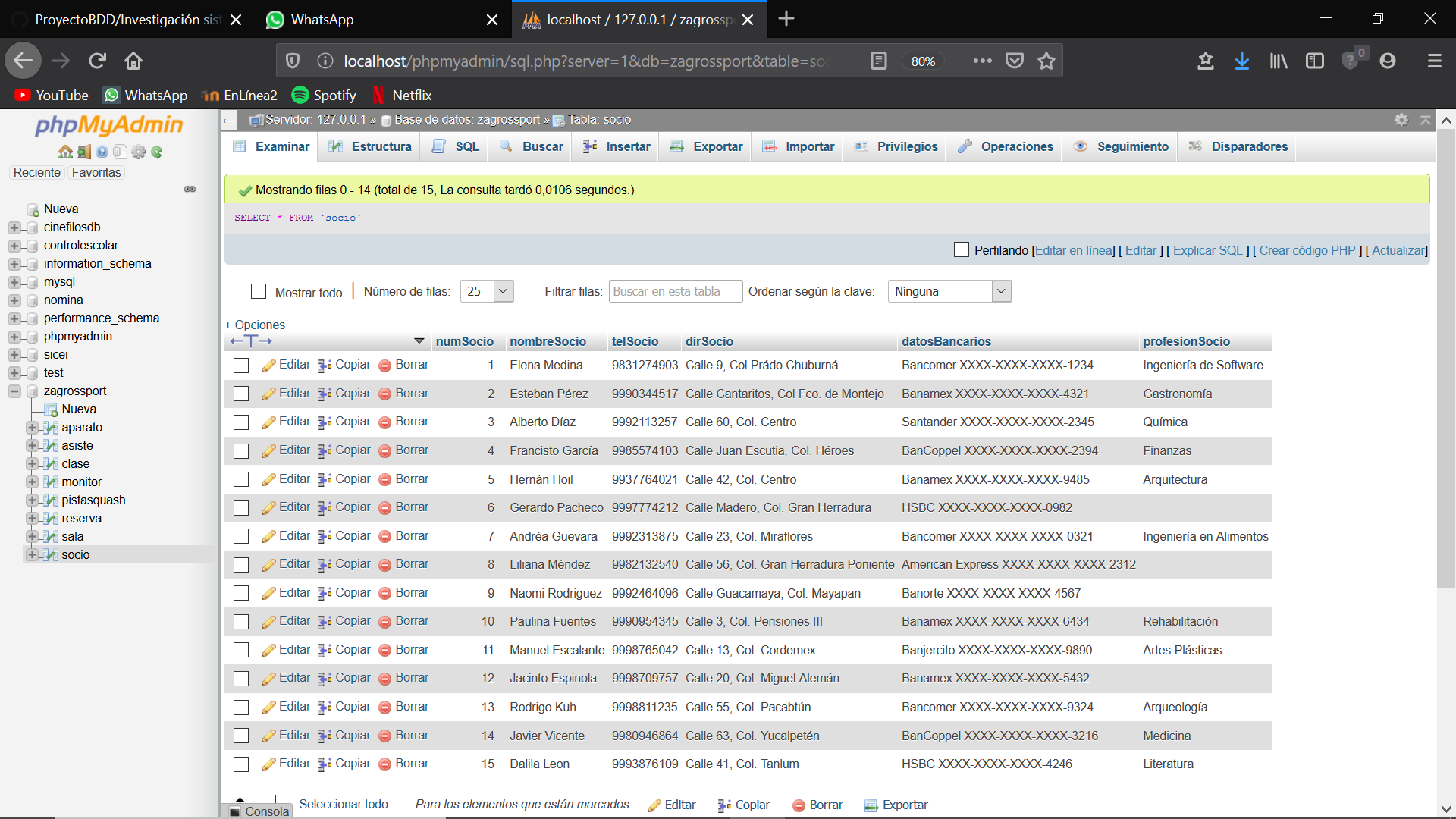
**Tabla Pista de Squash**

INSERT INTO `pistasquash` (`numPista`, `ubicacionPista`, `estadoPista`) VALUES ('1', 'Planta Baja, Campo 1', 'Bueno'), ('2', 'Planta Baja, Campo 2', 'Bueno'), ('3', 'Planta Baja, Campo 3', 'Excelente'), ('4', 'Planta Baja, Campo 4', 'Medio'), ('5', 'Planta Baja, Campo 5', 'Excelente');



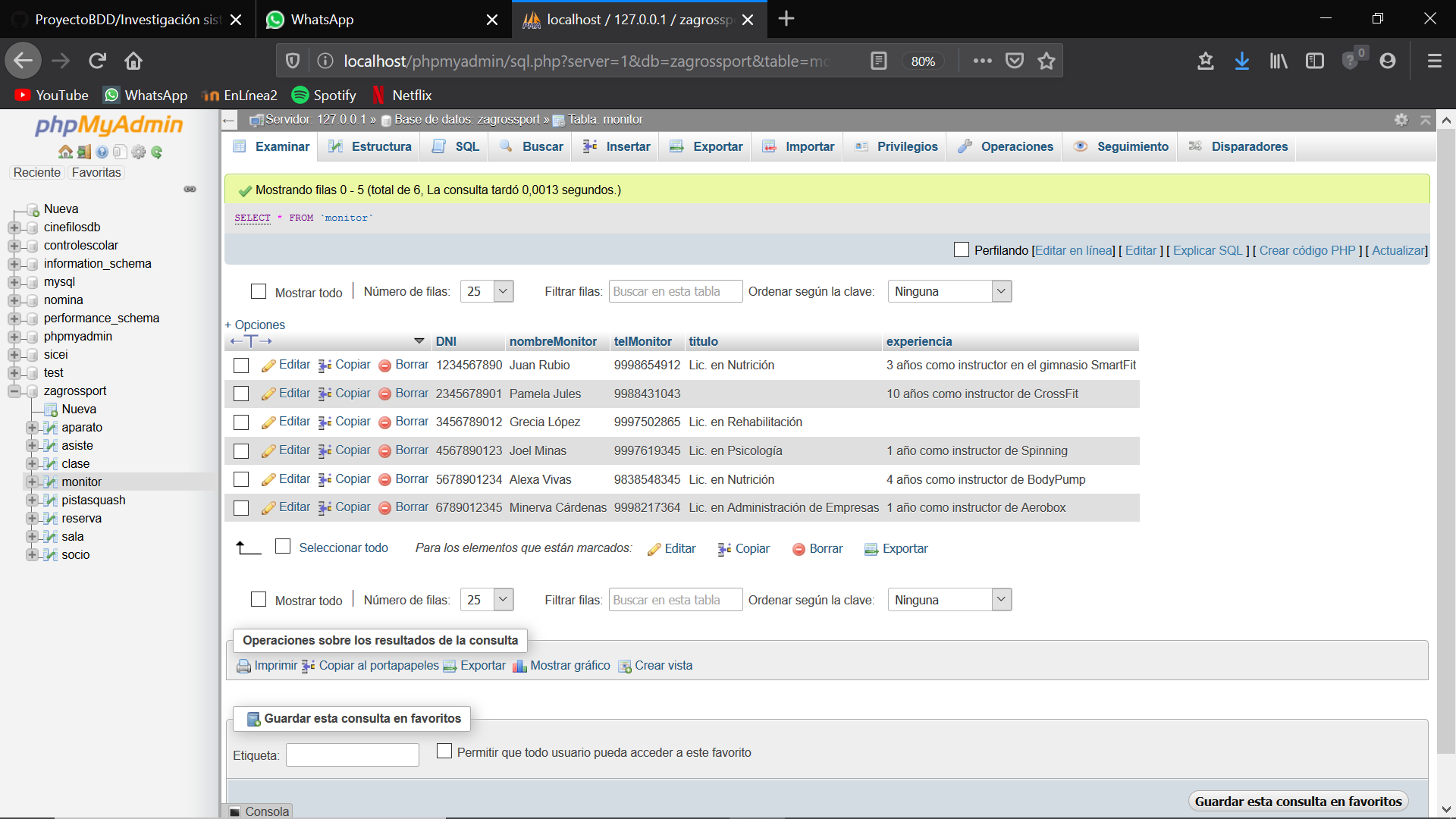
**Tabla Socio**

INSERT INTO `socio` (`numSocio`, `nombreSocio`, `telSocio`, `dirSocio`, `datosBancarios`, `profesionSocio`) VALUES ('6', 'Gerardo Pacheco', '9997774212', 'Calle Madero, Col. Gran Herradura', 'HSBC XXXX-XXXX-XXXX-0982', ''), ('7', 'Andréa Guevara', '9992313875', 'Calle 23, Col. Miraflores', 'Bancomer XXXX-XXXX-XXXX-0321', 'Lic. en Ingeniería en Alimentos'), ('8', 'Liliana Méndez', '9982132540', 'Calle 56, Col. Gran Herradura Poniente', 'American Express XXXX-XXXX-XXXX-2312', 'Lic. en Rehabilitación'), ('9', 'Naomi Rodriguez', '9992464096', 'Calle Guacamaya, Col. Mayapan', 'Banorte XXXX-XXXX-XXXX-4567', ''), ('10', 'Paulina Fuentes', '9990954345', 'Calle 3, Col. Pensiones III', 'Banamex XXXX-XXXX-XXXX-6434', '');



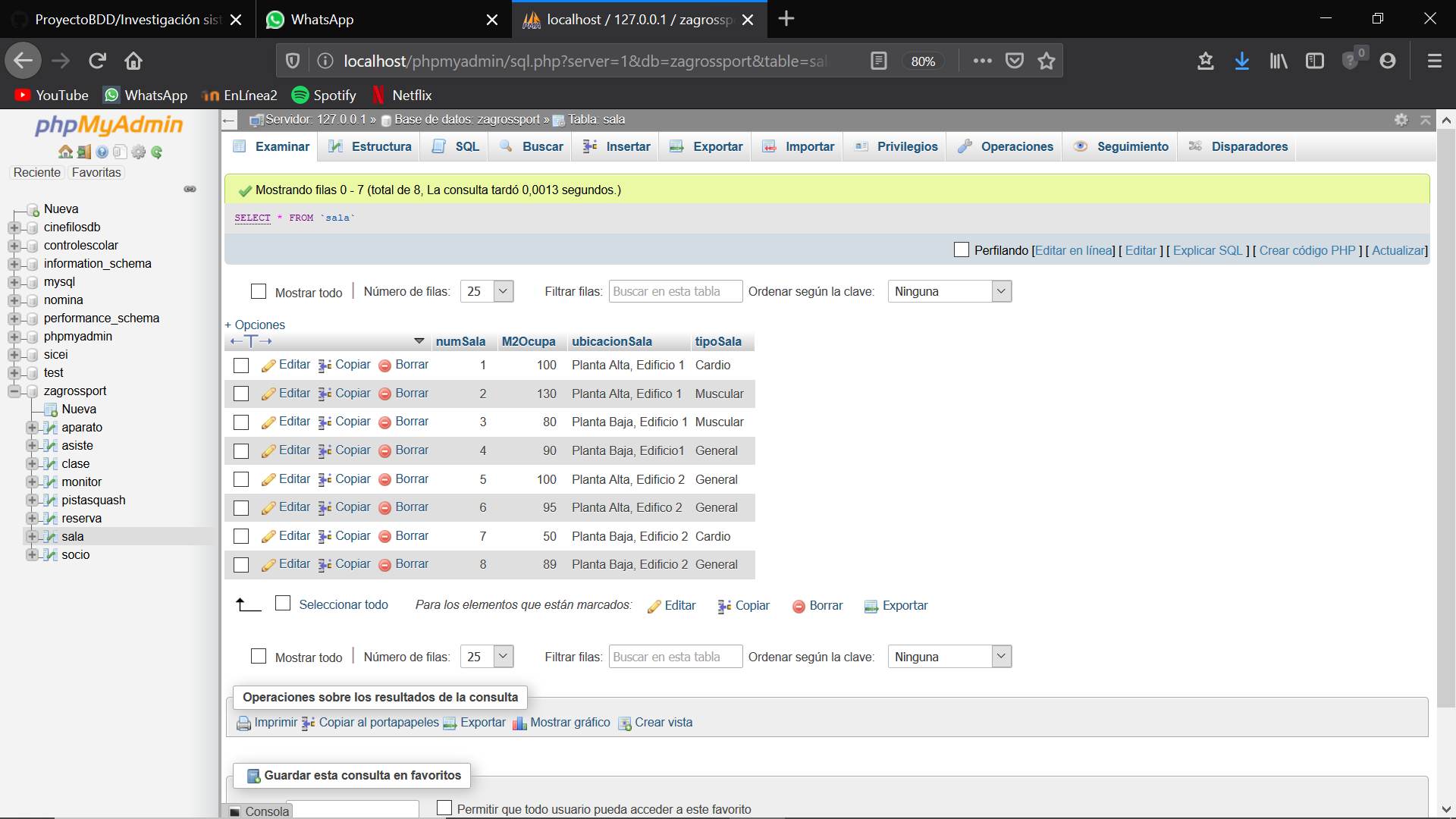
**Tabla Monitor**

INSERT INTO `monitor` (`DNI`, `nombreMonitor`, `telMonitor`, `titulo`, `experiencia`) VALUES ('1234567890', 'Juan Rubio', '9998654912', 'Lic. en Nutrición', '3 años como instructor en el gimnasio SmartFit'), ('2345678901', 'Pamela Jules', '9988431043', '', '10 años como instructor de CrossFit'), ('3456789012', 'Grecia López', '9997502865', 'Lic. en Rehabilitación', '');



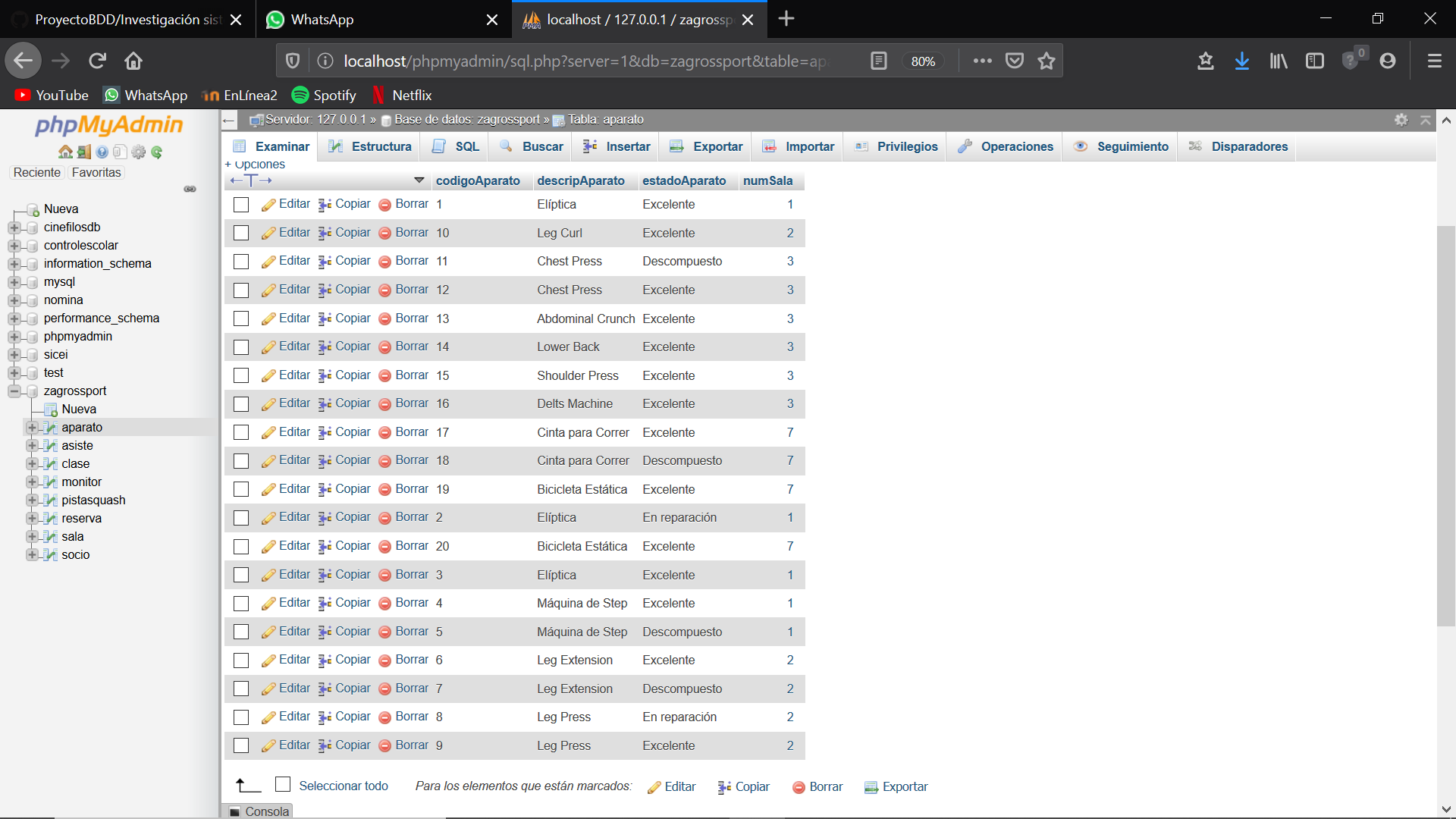
**Tabla Sala**

INSERT INTO `sala` (`numSala`, `M2Ocupa`, `ubicacionSala`, `tipoSala`) VALUES ('5', '100', 'Planta Alta, Edificio 2', 'General'), ('6', '95', 'Planta Alta, Edifico 2', 'General'), ('7', '50', 'Planta Baja, Edificio 2', 'Cardio'), ('8', '89', 'Planta Baja, Edificio 2', 'General');



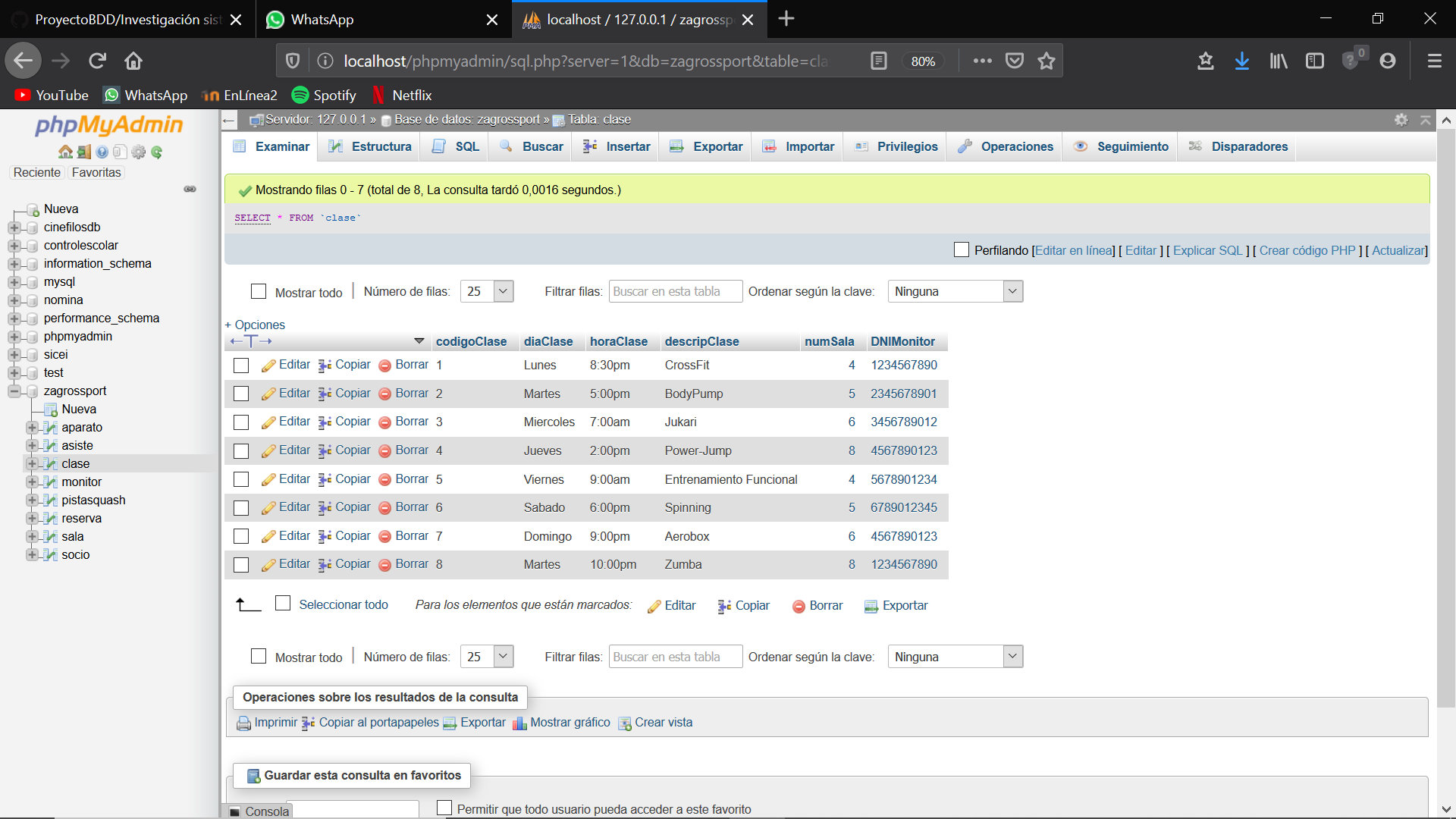
**Tabla Aparato**

INSERT INTO `aparato` (`codigoAparato`, `descripAparato`, `estadoAparado`, `numSala`) VALUES ('1', 'Elíptica', 'Excelente', '1'), ('2', 'Elíptica', 'Bueno', '1'), ('3', 'Elíptica', 'Excelente', '1'), ('4', 'Máquina de Step', 'Excelente', '1'), ('5', 'Máquina de Step', 'Descompuesto', '1');



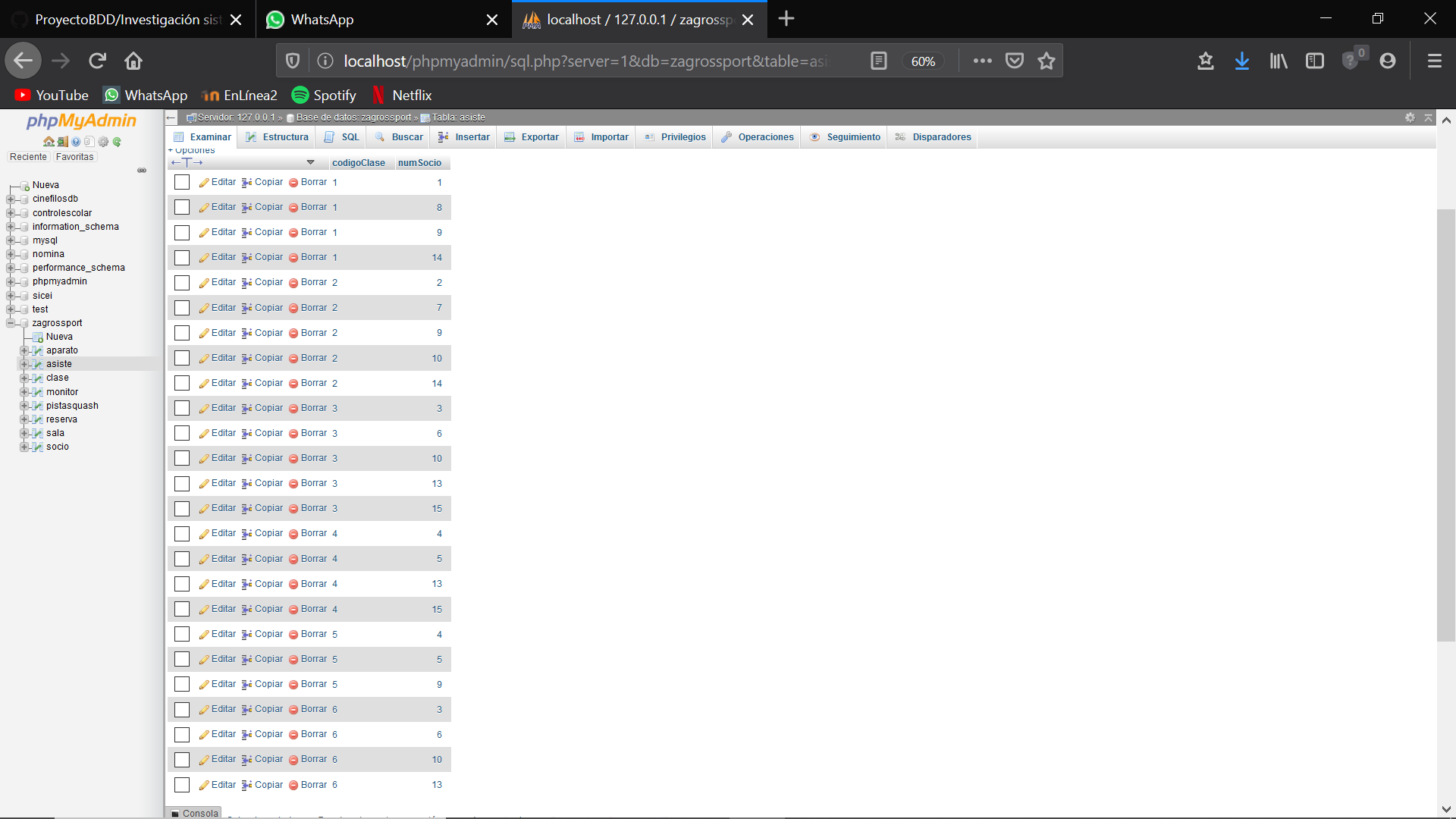
**Tabla Clase**

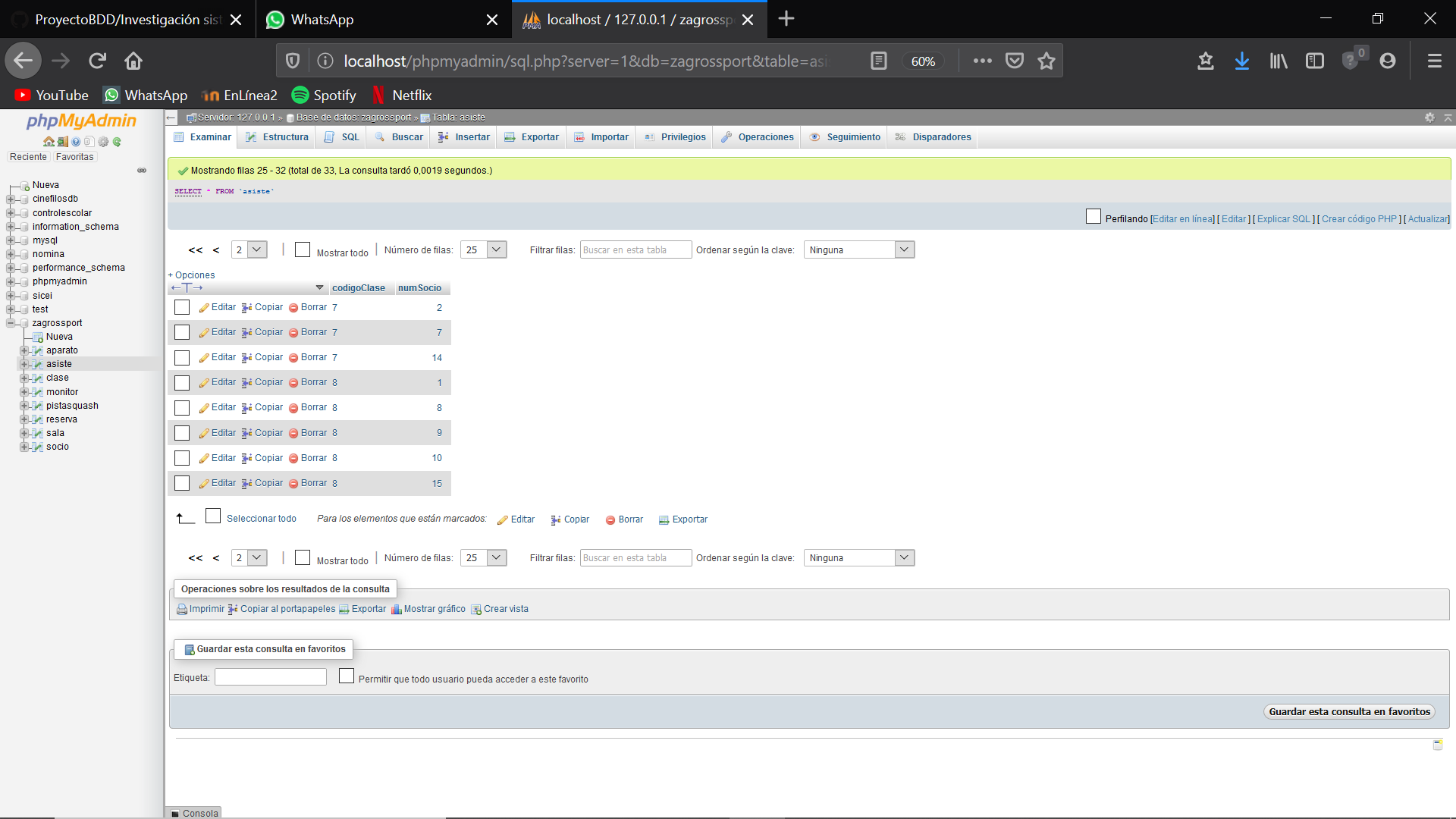
INSERT INTO `clase` (`codigoClase`, `diaClase`, `horaClase`, `descripClase`, `numSala`, `DNIMonitor`) VALUES ('5', 'Viernes', '9:00am', 'Entrenamiento Funcional', '4', '5678901234'), ('6', 'Sabado', '6:00pm', 'Spinning', '5', '6789012345'), ('7', 'Domingo', '9:00pm', 'Aerobox', '6', '4567890123'), ('8', 'Martes', '10:00pm', 'Zumba', '8', '1234567890');



**Tabla Asiste**

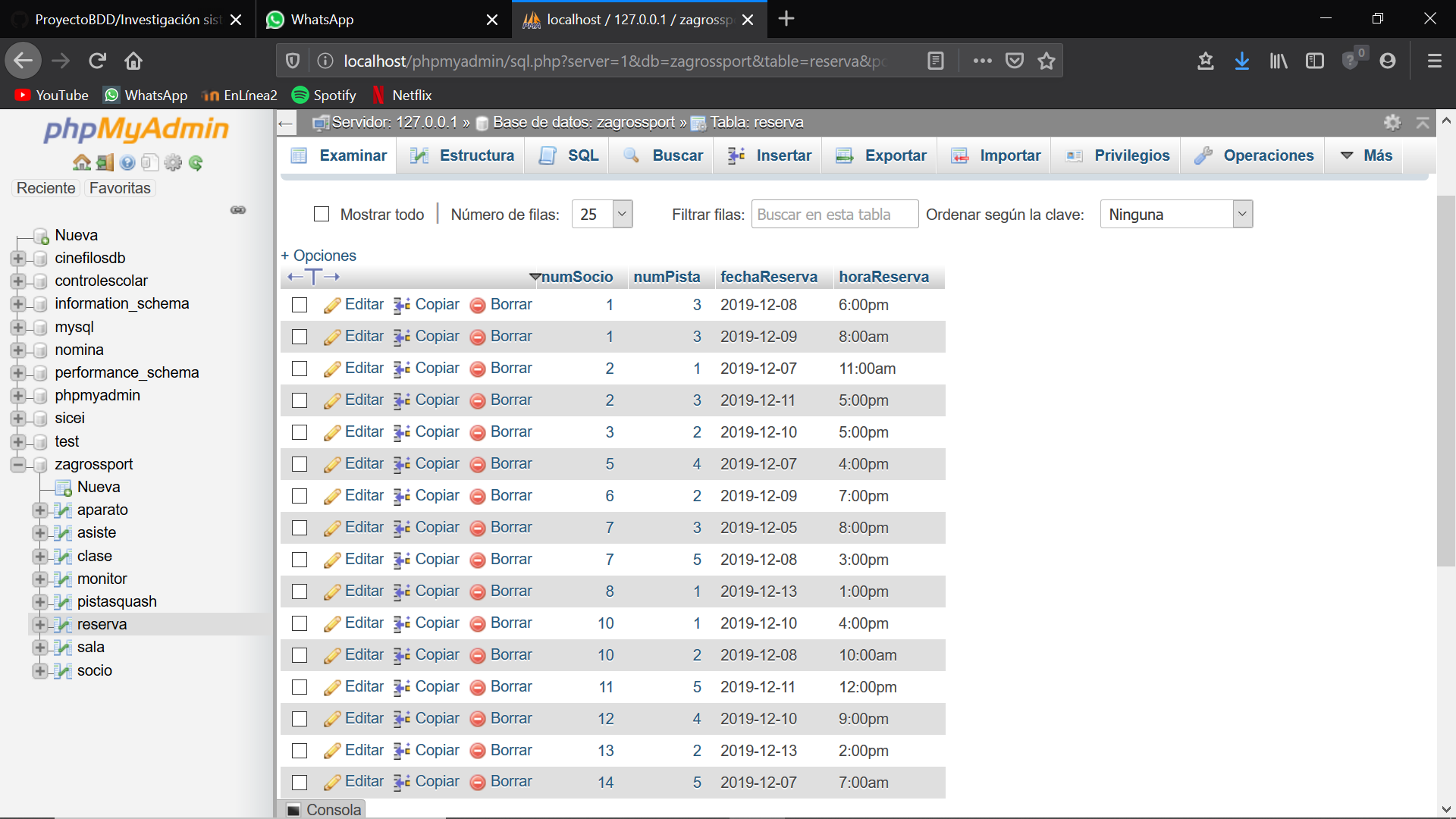
INSERT INTO `asiste` (`codigoClase`, `numSocio`) VALUES ('1', '1'), ('2', '2'), ('3', '3'), ('4', '4'), ('5', '5'), ('6', '6'), ('7', '7'), ('8', '8');





**Tabla Reserva**

INSERT INTO `reserva` (`numSocio`, `numPista`, `fechaReserva`, `horaReserva`) VALUES ('3', '2', '2019-12-10', '5:00pm'), ('1', '3', '2019-12-09', '8:00am'), ('10', '1', '2019-12-10', '4:00pm'), ('13', '2', '2019-12-13', '2:00pm');

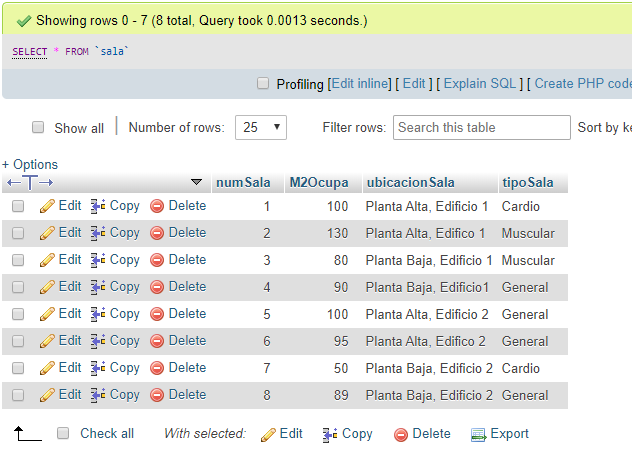


**Nota:** Estas son solo sentencias de ejemplo, los datos que aparecen en ellas si fueron ingresados a la BD pero no fueron los únicos, también se agregaron más. En las imágenes se pueden ver todos los datos que se ingresaron a las tablas.

VII. Utilizando el Sistema Manejador de Bases de Datos seleccionado, implemente el siguiente conjunto de consultas (incluya en el reporte, las sentencias utilizadas para las 7 solicitudes, así como capturas de pantalla de lo generado una vez ejecutados dichas consultas).

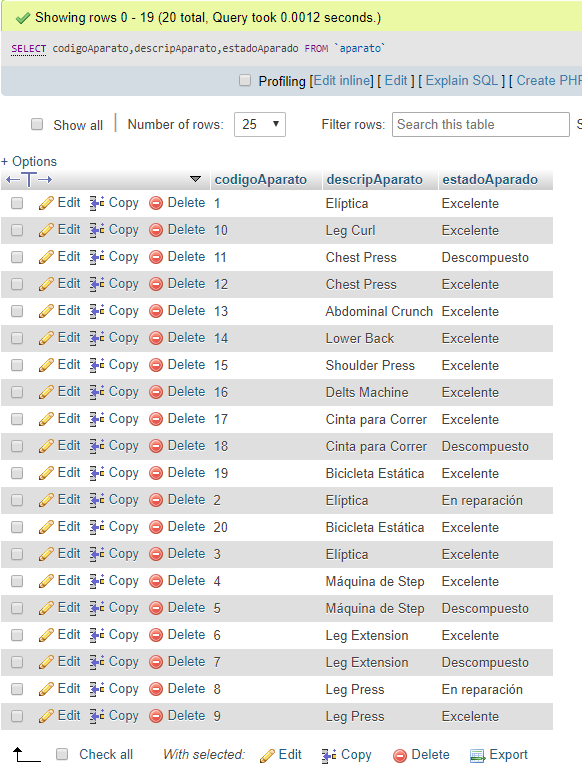
- Obtener la relación de salas de que consta el GYM, con el detalle: Identificador, Ubicación, Tipo, metros cuadrados.

**SELECT \* FROM `sala`**



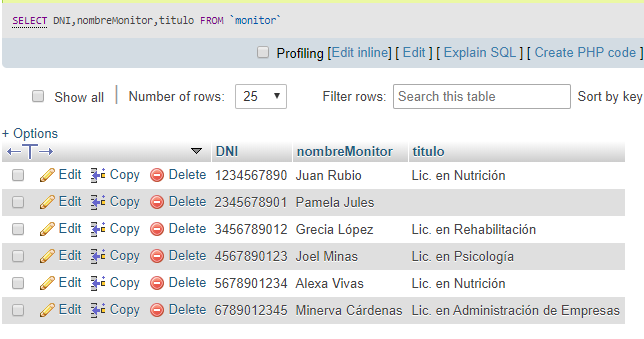
- Obtener la relación de aparatos adquiridos por el GYM con el detalle: Código, descripción, edo. Conservación.

**SELECT codigoAparato,descripAparato,estadoAparado FROM `aparato`**



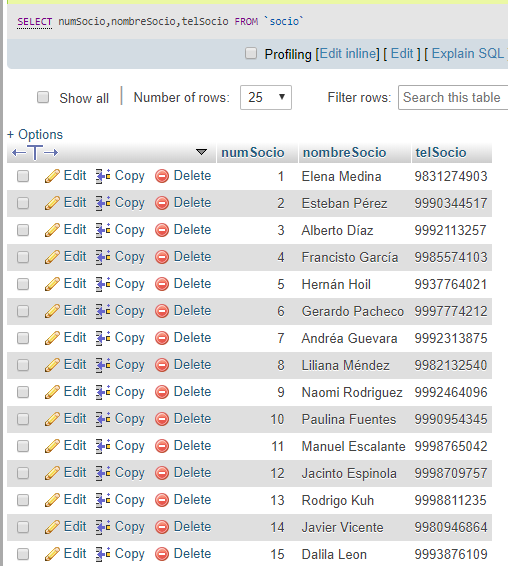
- Obtener la relación de instructores que trabajan para el GYM, con el detalle: DNI, Nombre, Preparación.

**SELECT DNI,nombreMonitor,titulo FROM `monitor`**



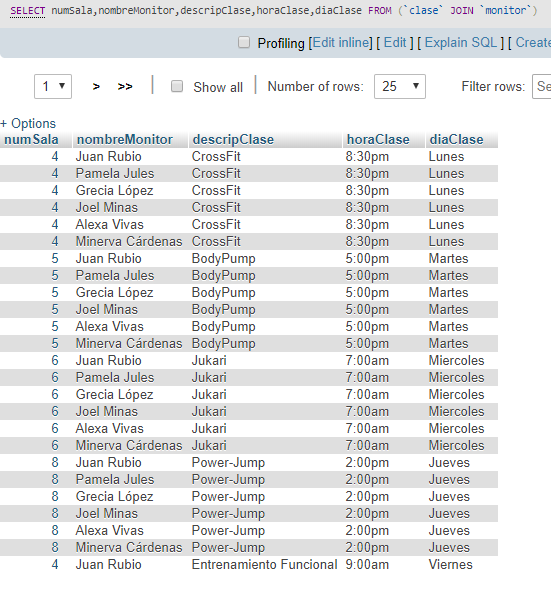
- Obtener la relación de socios registrados al GYM, con el detalle: No. Socio, Nombre, Teléfono.

**SELECT numSocio,nombreSocio,telSocio FROM `socio`**



- Obtener la relación de clases programadas para la semana, con el detalle: No. Sala, Instructor, Descripción, Horario, Día.

**SELECT numSala,nombreMonitor,descripClase,horaClase,diaClase FROM (`clase` JOIN `monitor`)**

****

- Obtener la relación de pistas reservadas para el fin de semana, con el detalle: Fecha, Hora, No. pista, No. Socio que realizó la reserva.

**SELECT \* FROM reserva WHERE fechaReserva BETWEEN ‘2019-12-6’ and ‘2019-12-8’**



-Agregue una consulta que considere requerida para el administrador del GYM (justifíquela).

**Relación de los aparatos (código, descripción y estado) cuyo estado sea descompuesto**

**Consideramos que esto es de suma relevancia debido a que estos aparatos pueden llegar a afectar el negocio, por tanto, es ser conscientes del estado de los aparatos, puede llegar a prevenir un posible accidente, y en caso de que se encuentren descompuestos, ayudar a localizar aquellos que requieran mantenimiento.**

**SELECT codigoAparato,descripAparato,estadoAparado FROM `aparato` WHERE estadoAparado='Descompuesto'**

