Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática

CI-1312 Bases de Datos I

Grupo 02

Prof. Elzbieta Malinowski

Documentación del
Proyecto de Base de Datos Relacional

Triton Multisport Database

Versión 1

Elaborado por:
Daniel Escamilla
Jose Mejías
Andrés Solís

CRÉDITOS: ***Plantilla original creada por la Prof. Gabriela Salazar. Modificaciones por la Prof. Alexandra Martínez y Prof. Elzbieta Malinowski para el curso CI-1312.***

I Semestre 2018

Tabla de contenido

ETAPA 1: Especificación de requerimientos y diseño conceptual de la base de datos

1. Introducción

1.1 Ámbito y alcance del Sistema

TritonMultiSport es una empresa con una plataforma *online* para deportistas de todos los perfiles: desde recreativos hasta élites. Dicha plataforma brinda programas semanales de entrenamiento con la asesoría de un experto. Además, hay un acompañamiento del progreso del atleta. El cliente puede escoger días de la semana para el entrenamiento (máximo seis) y uno o más deportes para realizar los entrenamientos. Hay dos métodos de cobro: individual o mensual. El individual se basa en comprar semanas de entrenamiento por separado, el mensual funciona como una suscripción el cual se va cobrando un monto determinado cada mes.

La base de datos es capaz de recorrer toda los datos recolectada por atleta. Así, la base de datos tiene los datos personales de los atletas en el sistema (nombre, correo, número telefónico, sexo, entre otros), los datos de la condición física (presión sanguínea, medidas corporales, entre otros), el tipo de cobro con toda su información necesaria (montos, posibles descuentos, fecha, etc) y los programas de entrenamientos.

1.2 Usuarios del Sistema

Grupo de Usuarios	**Características**	**Datos que requiere acceder**	**Operaciones que requiere ejecutar**
Atleta	Son atletas y su principal función en la base de datos es recibir bloques de entrenamiento y modificar sus datos personales.	Datos médicos. Presión sanguínea. Test cardiaco. Fechas de ingreso de datos de condición física. Datos de los bloques de entrenamiento. Datos de cobros. Datos personales.	Consultas sobre datos personales. Modificaciones periódicas a los datos de condición física y en cualquier momento a la información médica. Llenar un *feedback*

Entrenador	Los entrenadores tienen la responsabilidad de desarrollar bloques de entrenamiento especializados hasta cierto punto, con base en los datos personales ingresados por el atleta, además de los datos que se concluyen a partir de estos.	Datos de condición física de clientes y personales de los atletas. Bloques de entrenamiento. Nivel de cada deporte. Puede ver etiquetas de entrenamiento.	Consultas sobre datos personales y médicos de los atletas. Ingreso de bloques de entrenamiento. Actualizar nivel del atleta por deporte. Agregar etiquetas por entrenamiento.
Administrador	Es el encargado de la administración de los recursos del sistema de base de datos. Ingresa, modifica y retira cualquier dato en la base. Responsable de la seguridad de la base de datos, y del monitoreo de su uso.	Todos los datos disponibles en la base.	Edición, ingreso, retiro y consulta de todos los datos disponibles.

1.3 Definiciones, acrónimos y abstracciones

CVC: Código de verificación de la tarjeta.

ID: Identificador

2 Especificación de Requerimientos

2.1 Requerimientos de datos y restricciones

Nombre de posible	**Posibles atributos**	**Restricciones identificadas**	**Descripción**	**Ejemplos de instancias**

tipo de entidad o relación				
Atleta	nombre, apellido1, apellido2, sexo, fecha nacimiento, ciudad, provincia, país, núm. tel, correo, redes sociales, talla camisa, modelo/marca monitor cardíaco, contraseña cuenta		representa los datos personales y perfil deportivo del cliente	Nombre completo: Juan Santamaría Peraza. Sexo: Masculino. Fecha de nacimiento: 26/1/0000. Ciudad: Escazú. Provincia: San José. País: Costa Rica. Número telefónico: +506 8777-7777. Correo electrónico: juan@e.com. Redes sociales: http://facebook.com/jsantamaria. Talla de camisa: M. Marca de monitor cardíaco: RadioShack. Contraseña de perfil: juanito123.
Condición Física	información médica, condición cardíaca, mediciones funcionales, presión sanguínea, bioimpedancia,	todos los datos deben ser ingresados simultáneamente,	representa el estado de salud y condición deportiva del atleta, incluyendo una descripción de sus zonas cardiacas y datos biométricos	Condición física de Juan Santamaría en 24/05/2015. Información médica. 14-16, 17-20-21-25 Conclusiones de tests.

	identificador que incluye fecha de los datos y correo.			Valores de bioimpedancia. 8/8/2017.
Bloque Entrenamiento	rutina, etiqueta, día de inicio, días, horas, formulario, descripción y número de entrenamiento	cantidad de días debe ser igual o menor que 6, la descripción del bloque sólo es accesible por el entrenador	representa una unidad semanal del plan de entrenamiento del atleta, determinado por la condición física del atleta, y el horario y deportes seleccionados	Numero de entrenamiento 1. Etiqueta: Iron man. Día de inicio: 01-01-2017. Horas: 2.
Deporte	nombre, nivel, número de días, días a entrenar	el número de días a entrenar por deporte no debe superar 6	el atleta podrá escoger entrenar uno o más deportes entre natación, ciclismo, atletismo y acondicionamiento	Nombre: Natación. Nivel: intermedio. Número de días: 3. Días a entrenar: Lunes, Martes y Miércoles.
Cobro mensual	descuento[código, duración, porcentaje], monto mensual, monto final. fecha de pago, fecha de finalización, tarjeta[cvc, vencimiento, número] y ID factura	debe existir un límite para la frecuencia en el cambio de la fecha de pago	es un plan mensual de cobro automático, exime al cliente (atleta) del pago por inscripción, se realiza por medio electrónico con una tarjeta de crédito o débito	Código de descuento: 001. Duración de descuento: 12 meses. Porcentaje: 30%. Monto mensual: 5USD. Monto final: 2 USD. Fecha de pago: 30. Fecha de finalización: 23/8/2020. CVC de tarjeta: 123. Vencimiento de tarjeta: 8/8/8.

				Número de tarjeta: 1. Número de factura: 01.
Cobro individual	descuento[código, duración, porcentaje], monto semanal, monto total, monto de inscripción, cantidad de semanas, fecha de pago, fecha de inicialización, tarjeta[cvc, vencimiento, número] y ID factura		el atleta realiza pagos manuales por bloque de entrenamientos, y un único pago por inscripción al registrarse. se realiza por medio electrónico con una tarjeta de crédito o débito	en la fecha dd/mm/aa se hizo el pago por monto $X con la tarjeta XXXX, asociada al atleta AA
(Atleta)**Tiene**(Condición Física)		Un atleta solamente puede estar relacionado con una instancia de condición física. Todas las instancias de atleta deben estar relacionadas con al menos una instancia de condición física.	Relación que indica cómo se asocia un atleta con las condiciones físicas actual e históricas.	El atleta A tiene un listado de condiciones físicas entre las que se encuentra la actual.

(Bloque de Entrenamiento) **Determina**(Deporte)		un bloque de entrenamiento no puede incluir más que un deporte por cada día de entrenamiento	un bloque de entrenamiento puede incluir varios deportes en un solo bloque, pero solo un deporte por día de entrenamiento	el bloque B1 incluye ejercicios para el entrenamiento en ciclismo y natación para un atleta A de nivel avanzado
(Atleta) **Paga** (Cobro)		el atleta debe hacer sus pagos usando uno y solo uno de los planes de cobro, por cada pago	el atleta puede escoger un plan de pago, entre el mensual (de cobro automático) y el semanal (de cobro manual). el pago lo realiza por medio electrónico con una tarjeta de crédito o débito. se mantiene un historial de pagos para cada atleta	el atleta X realizo un pago por $Y en la fecha dd/mm/aa
(Condición física)**Determina**(Bloque de entrenamiento)		Una condición física puede estar relacionada con uno o más bloques de entrenamiento mientras que los bloques solo pueden estar relacionados con una condición física.	Dependiendo de la condición física del atleta se le asigna un bloque de entrenamiento específico condicionado por el conjunto de características de su condición física como los siguientes: historial médico, resultados cardíacos, etc.	Para una condición física X que posee ciertas características, los entrenadores se encargan de asignar un bloque de entrenamiento desarrollado en función de las cualidades y restricciones de dicha condición física.

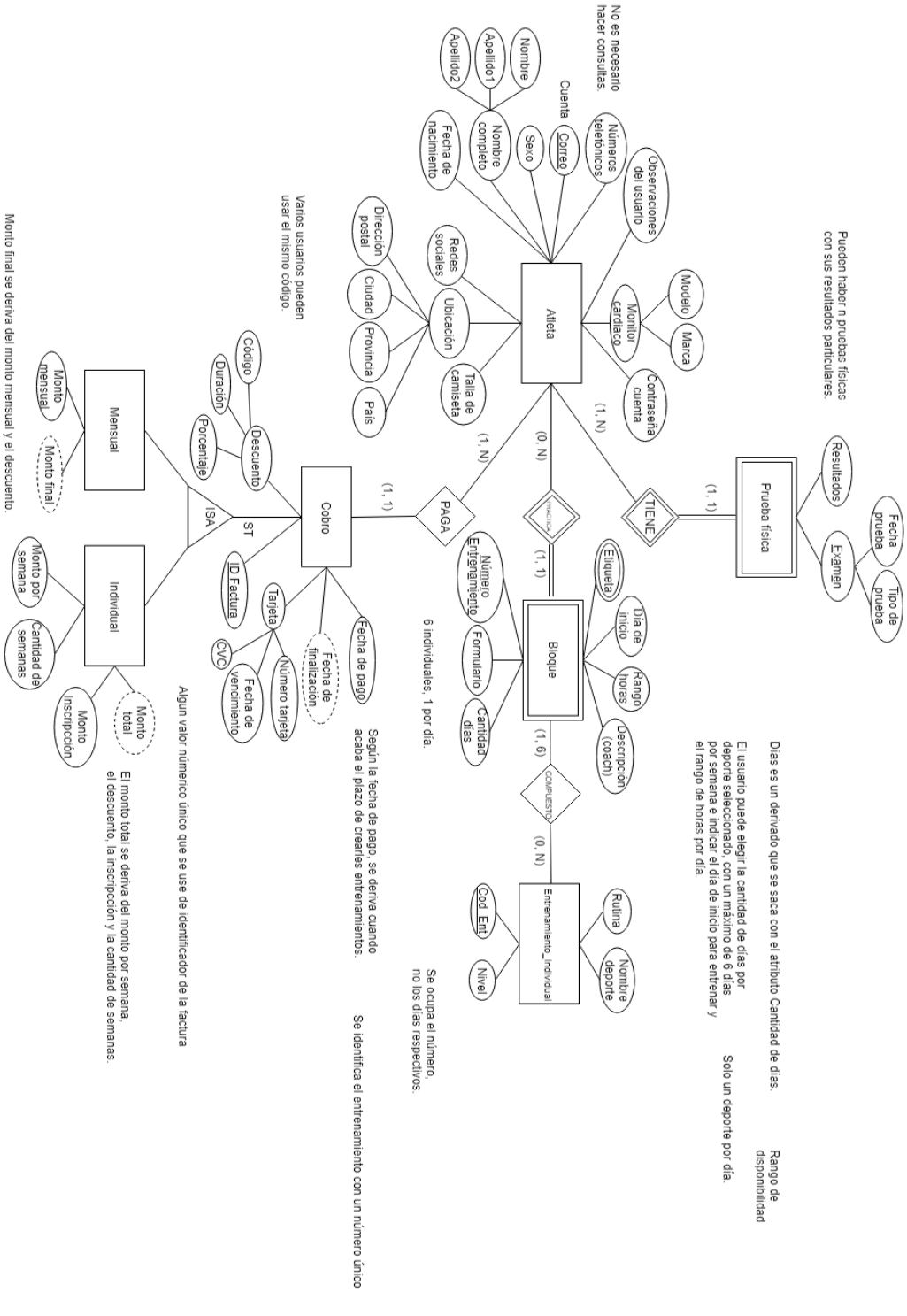
Figura 1. Ejemplos de descripción de datos requeridos por el usuario.

2.2 Requerimientos funcionales

1. Atleta consulta su bloque de entrenamiento.
2. Bloque de entrenamiento consulta datos de la condición física.
3. Atleta consulta los datos de la condición física.
4. Atleta consulta los cobros respectivos.
5. Consulta de los datos personales.
6. Cobro preguntas cuántos entrenamientos.
7. Qué deportes tiene un bloque de entrenamiento.

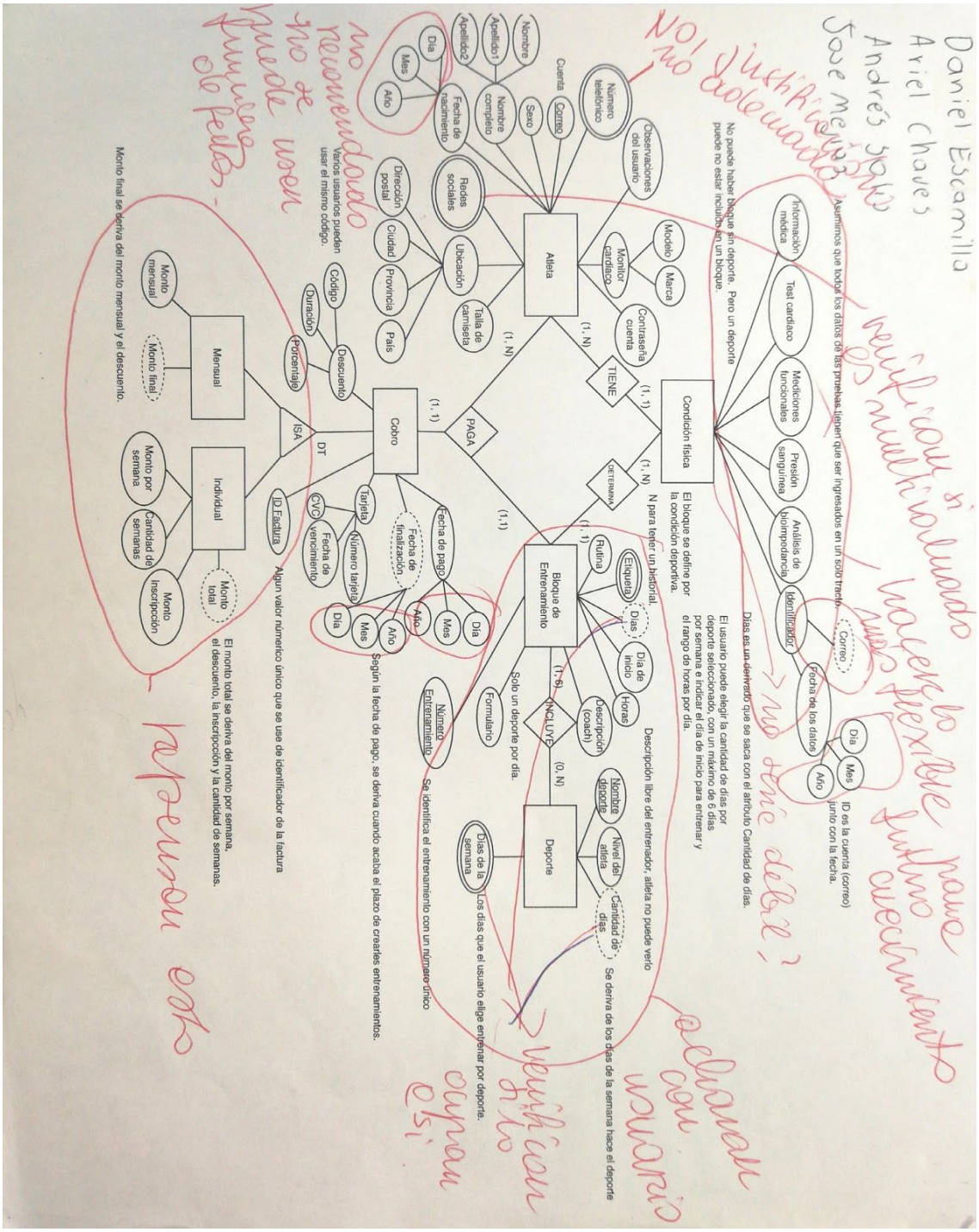
3 Diseño Conceptual de la Base de Datos

3.1 Esquema conceptual original de la BD

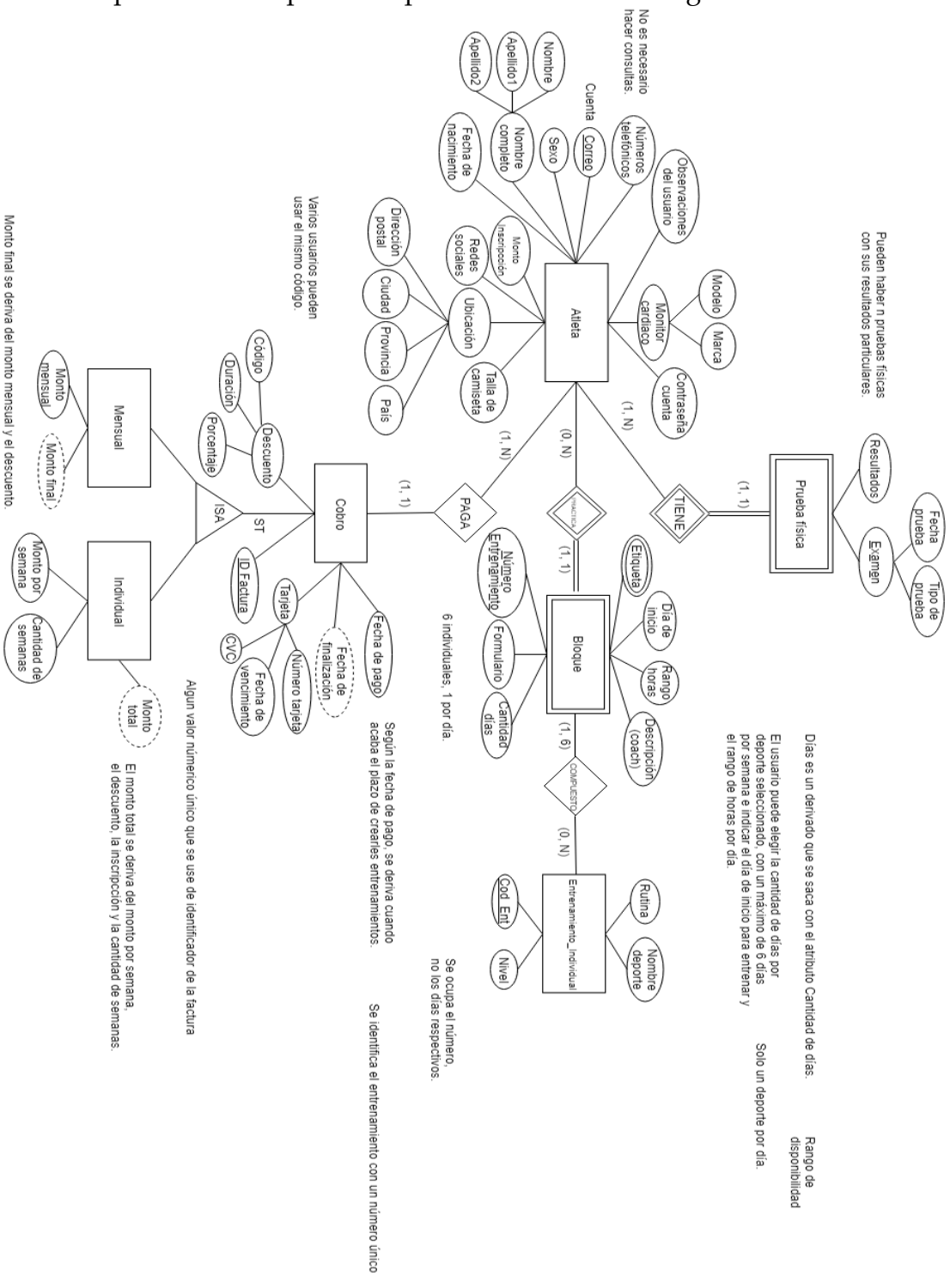


3.2 Esquema conceptual actualizado de la BD

Imagen del esquema original con los apuntes de los posibles cambios que se anotaron durante la defensa del diseño:



- Esquema nuevo que corresponde a los cambios sugeridos.

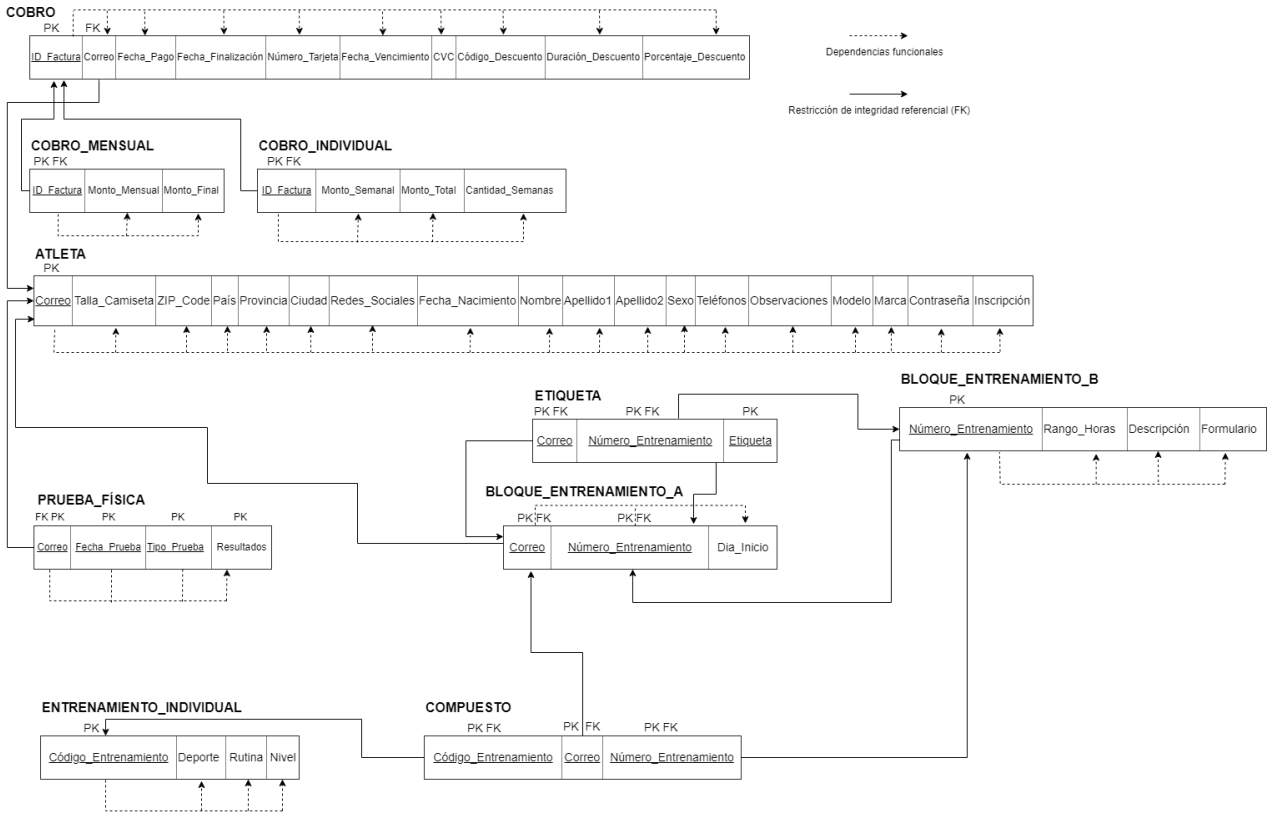


ETAPA 2: Diseño lógico y restricciones de la base de datos

4 Diseño Lógico de Base de Datos

4.1 Mapeo del esquema conceptual al esquema lógico relacional

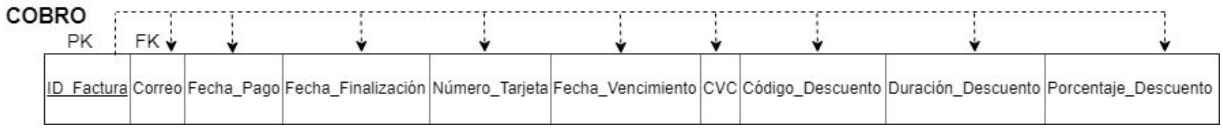
La siguiente figura es el estado final del mapeo luego de revisar las formas normales.

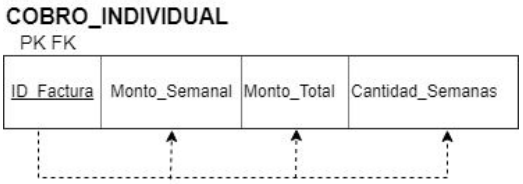


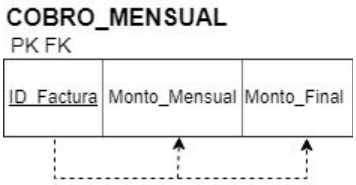
En el mapeo del esquema se tenía había un solo ISA, COBRO. Al ser COBRO un ISA solapado, solo se puede mapear usando las formas 8A y 8D. Se decidió usar el 8A porque, a diferencia del 8D, este no genera valores nulos para ningún atributo.

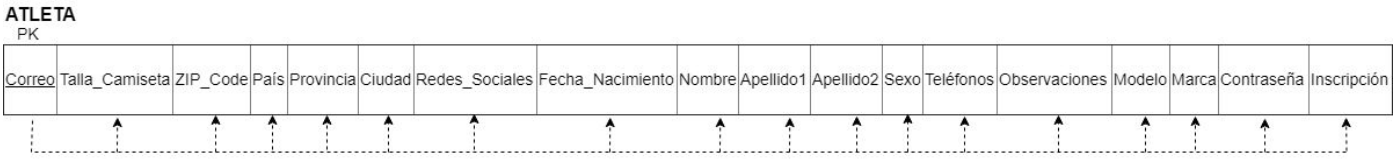
4.2 Verificación de la tercera forma normal

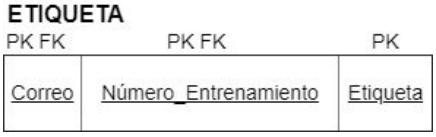
A continuación se presentan las tablas relacionales individuales de la base de datos. Se indica con flechas las dependencias funcionales en cada una. Se determina que cada tabla se encuentra en 3FN, luego de realizar las modificaciones pertinentes que así lo puedan asegurar, en cuyo caso se explica el proceso de separación.



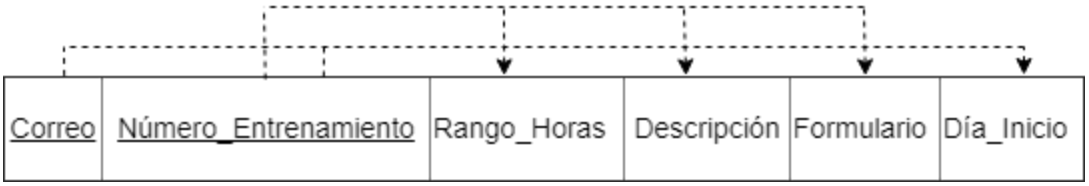






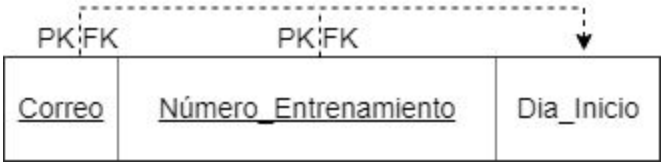


BLOQUE_ENTRENAMIENTO

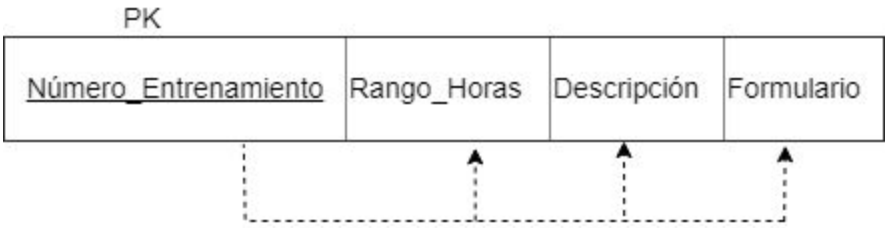


La anterior tabla, **BLOQUE_ENTRENAMIENTO,** no cumple con la 2FN, ya que Número_Entrenamiento→{Rango_Horas, Descripción, Formulario} es una dependencia funcional parcial. A su vez, la tabla tampoco cumple con la 3FN. Una vez aplicado sobre esta el procedimiento de normalización adecuado, es decir, descomponer la relación, resultaron las siguientes dos tablas normalizadas hasta la 3FN.

BLOQUE_ENTRENAMIENTO_A



BLOQUE_ENTRENAMIENTO_B



4.3 Especificación de tablas con restricciones

A partir de las tablas, se crearon los siguientes cuadros. Cada uno de estos cuadros es el nombre de alguna de las tablas con sus atributos. Para cada atributo se define el tipo de datos, la longitud o al grado de precisión y las restricciones.

Para el momento las restricciones agregadas con siglas son PK(*Primary Key*), FK (*Foreign Key*) y NN (*Not null*).

COBRO				
Nombre del atributo	**Tipo de datos**	**Longitud, precisión o rango de valores**	**Restricciones**	**Justificación (si aplica)**
ID_Factura	int	Positivo	PK	
Correo	string	[5,30] rango	FK(Atleta)	
Fecha_Pago	fecha			
Fecha_Fin	fecha			
Num_Tarjeta	string	16	NN	
Fecha_Vence	fecha		NN	
CVC	int	3	NN	
Codigo_Desc	int	desconocido		
Duracion_Desc	fecha			
Porcentaje_Desc	int	<=100		

COBRO_MENSUAL				
Nombre del atributo	**Tipo de datos**	**Longitud, precisión o rango de valores**	**Restricciones**	**Justificación (si aplica)**
ID_Factura	int	De 1 al máximo de un int	PK FK(Cobro)	
Monto_Mensual	int	Positivo	NN	
Monto_Final	int	Positivo	NN	

COBRO_INDIVIDUAL				
Nombre del atributo	**Tipo de datos**	**Longitud, precisión o rango de valores**	**Restricciones**	**Justificación (si aplica)**
ID_Factura	int	De 1 al máximo de un int	PK FK(Cobro)	
Monto_Semanal	int	Positivo	NN	
Monto_Total	int	Positivo	NN	
Cantidad_Semanas	int	Positivo	NN	

ATLETA				
Nombre del atributo	**Tipo de datos**	**Longitud, precisión o rango de valores**	**Restricciones**	**Justificación (si aplica)**
Correo	char	Longitud máxima de un string.	PK	
Talla_Camiseta	char	4	XS|S|M|L|XL|XXL|XXXL	
ZIP_Code	int	5		
País	char	[2,30] rango	solo letras	
Provincia	char	[2,30] rango	solo letras	
Ciudad	char	[2,30] rango	solo letras	
Redes_Sociales	char	[2,30] rango		
Fecha_Nac	fecha			
Nombre	char	[2,50] rango	NN	
Apellido1	char	[2,50] rango	NN	
Apelllido2	char	[2,50] rango		
Sexo	char	1	M | F	
Teléfonos	int	20		
Observaciones	char	500		
Modelo	char	50		
Marca	char	50		
Contraseña	char	[8,20] rango	NN	
Inscripción	int	Valor positivo	NN	

BLOQUE_ENTRENAMIENTO_A				
Nombre del atributo	**Tipo de datos**	**Longitud, precisión o rango de valores**	**Restricciones**	**Justificación (si aplica)**
Correo	string	[5,30] rango	FK(Atleta)	
Día_Inicio	Fecha		NN	

BLOQUE_ENTRENAMIENTO_B				
Nombre del atributo	**Tipo de datos**	**Longitud, precisión o rango de valores**	**Restricciones**	**Justificación (si aplica)**
Num_Entren	int	Positivo	PK Fk (Bloque entrenamiento A)	
Rango_Horas	int	2	<24 NN	
Descripción	char	500		
Formulario	char	500		

ENTRENAMIENTO_INDIVIDUAL				
Nombre del atributo	**Tipo de datos**	**Longitud, precisión o rango de valores**	**Restricciones**	**Justificación (si aplica)**
Código_Entren	int	Positivo	PK	
Deporte	string	20	NN	
Rutina	string	Longitud indefinida	NN	
Nivel	string	12	Principiante | intermedio | avanzado NN	

COMPUESTO				
Nombre del atributo	**Tipo de datos**	**Longitud, precisión o rango de valores**	**Restricciones**	**Justificación (si aplica)**
Código_Entren	int	Positivo	PK FK (entrenamiento individual)	
Correo	string	[5,30] rango	PK FK(bloque entrenamiento A)	
Núm_Entren	int	Positivo	PK FK(bloque entrenamiento A)	

ETIQUETA				
Nombre del atributo	**Tipo de datos**	**Longitud, precisión o rango de valores**	**Restricciones**	**Justificación (si aplica)**
Correo	string	[5,30] rango	PK FK(bloque entrenamiento A)	
Núm_Entren	int	Positivo	PK FK(bloque entrenamiento A o B)	
Etiqueta	char	50	PK	

PRUEBA FÍSICA				
Nombre del atributo	**Tipo de datos**	**Longitud, precisión o rango de valores**	**Restricciones**	**Justificación (si aplica)**
Correo	string	[5,30]]rango	PK FK(Atleta)	
Fecha_Prueba	Fecha		PK	

Tipo_Prueba	char	50	PK	
Resultados	char	500	NN	

4.4 Revisión del esquema lógico con respecto a las operaciones

La siguiente tabla describe el camino que se debe seguir para responder a las operaciones más frecuentes en el sistema. Se indica la cantidad de tablas y operaciones *join* que se deben utilizar para realizar cada consulta.

CONSULTA FRECUENTE	CAMINO A SEGUIR	JOINS	CANTIDAD DE TABLAS
Consultar el bloque de entrenamiento de un atleta	Atleta→BloqueEntrenamientoA→Compuesto→EntrenamientoIndividual	Join(Atleta, Bloque Entrenamiento A, Compuesto, EntrenamientoIndividual)	4
Consultar los datos de prueba física de un atleta	Atleta→PruebaFisica	Join(Atleta, Prueba física)	2
Consultar el monto de pago(s) de un atleta	Atleta→Cobro->Mensual|Individual	Join(Atleta, Cobro, (Mensual|Individual))	2
Consultar los datos personales de un atleta	Atleta	No joins.	1
Consultar los deportes en un bloque de entrenamiento	BloqueEntrenamientoA→Compuesto→EntrenamientoIndividual	Join(BloqueEntrenamientoA, Compuesto, EntrenamientoIndividual)	3

ETAPA 3: Implementación de la base de datos

5 Especificaciones técnicas de la implementación

5.1 Características técnicas de la implementación

Para la implementación de la base de datos se utilizó al lenguaje de programación SQL. El DBMS en el que se trabajó fue en el Microsoft SQL Server Management System 17. No fue necesario ningún *driver* o herramienta extra además de las proporcionadas por la profesora. Para el proyecto se utilizó la arquitectura Cliente/Servidor.

5.2 Creación de tablas con restricciones

```
CREATE TABLE Atleta
(
	Codigo_Atleta_PK	INTEGER IDENTITY,	-- IDENTITY genera un valor único para cada tupla.  Por defecto inicia en 1.
	Correo			VARCHAR(30)	UNIQUE,
	Talla_Camiseta		VARCHAR(30),
	ZIP_Code		INTEGER,
	País			VARCHAR(30),
	Provincia		VARCHAR(30),
	Ciudad			VARCHAR(30),
	Redes_Sociales		VARCHAR(30),
	Fecha_Nacimiento	DATE,
	Nombre			VARCHAR(50)	NOT NULL,
	Apellido1		VARCHAR(50)	NOT NULL,
	Apellido2		VARCHAR(50),
	Sexo			CHAR,
	Telefonos		VARCHAR(20),	-- OJO.
	Observaciones		VARCHAR(500),
	Modelo			VARCHAR(50),
	Marca			VARCHAR(50),
	Contraseña		VARCHAR(20)	NOT NULL,
	Inscripcion		MONEY		NOT NULL

	CONSTRAINT PK_Codigo_Atleta PRIMARY KEY (Codigo_Atleta_PK),
	CONSTRAINT CK_ZIP_Code_Atleta CHECK (ZIP_Code >= 10000 AND ZIP_CODE <= 99999),  -- 5 dígitos.
	CONSTRAINT CK_Inscripcion_Atleta CHECK (Inscripcion >= 0)
);
```

```
CREATE TABLE Prueba_Fisica
(
	Codigo_Atleta_FK	INTEGER,
	Fecha_Prueba_PK		DATE,
	Tipo_Prueba_PK		VARCHAR(50),
	Resultados			VARCHAR(500) NOT NULL   -- Si se crea la tupla, obligatoriamente se hizo una prueba, por lo que t

	CONSTRAINT PK_Codigo_Fecha_Prueba_Tipo_Prueba_Prueba_Fisica PRIMARY KEY (Codigo_Atleta_FK, Fecha_Prueba_PK, Tipo_Prueba_PK ),
	CONSTRAINT FK_Codigo_Prueba_Fisica_Prueba_Fisica FOREIGN KEY (Codigo_Atleta_FK) REFERENCES Atleta(Codigo_Atleta_PK)
		ON DELETE	CASCADE -- Si se elimina el atleta, no hay porqué mantener las pruebas.
		ON UPDATE	CASCADE
);


CREATE TABLE Cobro
(
	ID_Factura_PK			INTEGER,
	Codigo_Atleta_FK		INTEGER DEFAULT -1,
	Fecha_Pago				DATE		NOT NULL,
	Fecha_Finalización		DATE		NOT NULL,
	Número_Tarjeta			VARCHAR(16)	NOT NULL,
	Fecha_Vencimiento		DATE		NOT NULL,
	CVC						INTEGER		NOT NULL,
	Codigo_Descuento		INTEGER,
	Duracion_Descuento		DATE,
	Porcentaje_Descuento	INTEGER DEFAULT 0	-- CREO JEJE

	CONSTRAINT PK_ID_Factura_Cobro PRIMARY KEY (ID_Factura_PK),
	CONSTRAINT FK_Correo_Cobro FOREIGN KEY (Codigo_Atleta_FK) REFERENCES Atleta(Codigo_Atleta_PK)
		ON DELETE SET DEFAULT
		ON UPDATE CASCADE,
	CONSTRAINT CK_CVC_Cobro CHECK( CVC <= 999 AND CVC >= 0 ),
	CONSTRAINT CK_Porcentaje_Descuento_Cobro CHECK( Porcentaje_Descuento >= 0 AND  Porcentaje_Descuento <= 100 )
);


CREATE TABLE Cobro_Mensual
(
	ID_Factura_FK	INTEGER,
	Monto_Mensual	MONEY	NOT NULL,
	Monto_Final		MONEY				-- Derivado

	CONSTRAINT PK_ID_Factura_Cobro_Mensual PRIMARY KEY (ID_Factura_FK),
	CONSTRAINT FK_ID_Factura_Cobro_Mensual FOREIGN KEY(ID_Factura_FK) REFERENCES Cobro(ID_Factura_PK)
		ON DELETE CASCADE
		ON UPDATE CASCADE,
	CONSTRAINT CK_Monto_Mensual_Cobro_Mensual CHECK( Monto_Mensual >= 0 ),
	CONSTRAINT CK_Monto_Final_Cobro_Mensual CHECK( Monto_Final >= 0 )
);
```

```
CREATE TABLE Cobro_Individual
(
        ID_Factura_FK           INTEGER,
        Monto_Semanal           MONEY   NOT NULL,
        Monto_Total                     MONEY,                          -- Derivado
        Cantidad_Semanas        INTEGER NOT NULL,

        CONSTRAINT PK_ID_Factura_Cobro_Individual PRIMARY KEY (ID_Factura_FK),
        CONSTRAINT FK_ID_Factura_Cobro_Individual FOREIGN KEY(ID_Factura_FK) REFERENCES Cobro(ID_Factura_PK)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        CONSTRAINT CK_Monto_Semanal_Cobro_Individual CHECK( Monto_Semanal > 0 ),
        CONSTRAINT CK_Monto_Total_Cobro_Individual CHECK( Monto_Total > 0 ),
        CONSTRAINT CK_Cantidad_Semanas_Cobro_Individual CHECK( Cantidad_Semanas > 0 )
);


CREATE TABLE Bloque_Entrenamiento
(
        Codigo_Atleta_FK                INTEGER,
        Numero_Entrenamiento_PK INTEGER,
        Rango_Horas                             FLOAT                   NOT NULL,
        Descripcion                             VARCHAR(500),
        Formulario                              VARCHAR(500),
        Dia_Inicio                              VARCHAR(10)             NOT NULL,

        CONSTRAINT PK_Codigo_Numero_Entrenamiento_Bloque_Entrenamiento PRIMARY KEY (Codigo_Atleta_FK, Numero_Entrenamiento_PK),
        CONSTRAINT FK_Codigo_Bloque_Entrenamiento FOREIGN KEY(Codigo_Atleta_FK) REFERENCES Atleta(Codigo_Atleta_PK)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        CONSTRAINT CK_Rango_Horas_Bloque_Entrenamiento CHECK (Rango_Horas < 24.0 AND Rango_Horas > 0.0)

);

CREATE TABLE Entrenamiento_Individual
(
        Codigo_Entrenamiento_PK INTEGER,
        Deporte                                 VARCHAR(20)             NOT NULL,
        Rutina                                  VARCHAR(500)    NOT NULL,
        Nivel                                   VARCHAR(12)             NOT NULL

        CONSTRAINT PK_Codigo_Entrenamiento_Entrenamiento_Individual PRIMARY KEY(Codigo_Entrenamiento_PK)
);
```

```
CREATE TABLE Compuesto
(
    Codigo_Entrenamiento_FK INTEGER,
    Codigo_Atleta_FK        INTEGER,
    Numero_Entrenamiento_FK INTEGER

    CONSTRAINT PK_Codigo_Entrenamiento_Codigo_Atleta_Numero_Entrenamiento_Compuesto PRIMARY KEY (Codigo_Entrenamiento_FK, Codigo_Atleta_FK, Numero_Entrenamiento_FK),
    CONSTRAINT FK_Codigo_Numero_Entrenamiento_Compuesto FOREIGN KEY (Codigo_Atleta_FK, Numero_Entrenamiento_FK) REFERENCES Bloque_Entrenamiento(Codigo_Atleta_FK, Numero_Entrenamiento_PK)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT FK_Codigo_Entrenamiento_Compuesto FOREIGN KEY (Codigo_Entrenamiento_FK) REFERENCES Entrenamiento_Individual(Codigo_Entrenamiento_PK)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE Etiqueta
(
    Etiqueta_PK             VARCHAR(50),
    Codigo_Entrenamiento_FK INTEGER

    CONSTRAINT PK_Correo_Numero_Entrenamiento_Etiqueta PRIMARY KEY (Etiqueta_PK, Codigo_Entrenamiento_FK),
    CONSTRAINT FK_Etiqueta_Codigo_Entrenamiento FOREIGN KEY (Codigo_Entrenamiento_FK) REFERENCES Entrenamiento_Individual(Codigo_Entrenamiento_PK)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

5.3 Programación de base de datos

Implementación de disparadores

A la hora de implementar los disparadores, se halló que la DBMS ejecuta el trigger una sola vez por inserción, es decir, *Inserted* contiene la cantidad de tuplas insertadas, por lo que, por ejemplo, si se quiere obtener el atributo de una tabla, a primera vista, se pensaría que es de la siguiente forma:

```
SET @Porcentaje =
    (SELECT COALESCE(c.Porcentaje_Descuento, 0)
    FROM Inserted i JOIN Cobro c ON c.ID_Factura_PK = i.ID_Factura_FK)
```

El problema es que este *Select*, al hacer una inserción de varias tuplas en una misma instrucción, devuelve más de un valor (específicamente la cantidad de tuplas), por lo que no es posible usar el *Select,* todo esto debido a la naturaleza de la DBMS. Por ende, la única forma es usar cursores.

Respecto al disparador, lo que hace es calcular el atributo derivado *Monto_Final* de la tabla *Cobro_Individual,* esto aplicando un descuento -cuando es necesario-, dichos datos se consiguen en la tabla padre *Cobro*, de ahí que el cursor hace un *Join.* Al final, se hace un *Uptade* a cada tupla, modificando dicho atributo.

```
-- Al insertar una tupla, calculo el Monto_Final
CREATE TRIGGER TR_Mensual_Monto_Final
ON Cobro_Mensual AFTER INSERT
AS
        DECLARE @Monto_Final    MONEY
        DECLARE @Monto_Mensual  MONEY
        DECLARE @id_factura             INTEGER
        DECLARE @Porcentaje             FLOAT

        DECLARE Cursor_Tupla_Individual CURSOR LOCAL FOR
                SELECT c.Porcentaje_Descuento, i.Monto_Mensual, i.ID_Factura_FK
                FROM INSERTED i JOIN Cobro c ON i.ID_Factura_FK = c.ID_Factura_PK;

                OPEN Cursor_Tupla_Individual
                FETCH NEXT FROM Cursor_Tupla_Individual INTO @Porcentaje, @Monto_Mensual, @id_factura

                WHILE (@@FETCH_STATUS <> -1)
                BEGIN

                        IF @Porcentaje IS NULL          -- Porcentaje puede ser nulo.
                                SET @Porcentaje = 0

                        SET @Monto_Final = @Monto_Mensual*( 1 - (@Porcentaje / 100) )

                        UPDATE Cobro_Mensual
                        SET Monto_Final = @Monto_Final
                        WHERE @id_factura = ID_Factura_FK

                        FETCH NEXT FROM Cursor_Tupla_Individual INTO @Porcentaje, @Monto_Mensual, @id_factura
                END
```

Implementación de los procedimiento(s) almacenados(s) y funciones

El procedimiento almacenado *sp_cantidad_pruebas_fisicas* devuelve el número de pruebas que han hecho los atletas, esto usando *Group by.* La función almacenada *Obtener_Ganancias_Mes* devuelve las ganancias de un mes y año en específico.

```
-- Contar cuántas pruebas físicas han hechos los atletas.

CREATE PROC sp_cantidad_pruebas_fisicas AS
        SELECT a.Correo_PK, a.Nombre + ' ' + a.Apellido1 + ' ' + a.Apellido2 AS 'Nombre', COUNT(p.Fecha_Prueba_PK)
        FROM Triton.dbo.Atleta a        JOIN Triton.dbo.Prueba_Fisica p ON a.Correo_PK = p.Correo_FK
        GROUP BY a.Correo_PK, a.Nombre, a.Apellido1, a.Apellido2
```

```
CREATE FUNCTION Obtener_Ganancias_Mes(@Mes TINYINT )
RETURNS INTEGER
AS
BEGIN
    DECLARE @Ganancias INTEGER

    IF  EXISTS  (
    SELECT Cobro.ID_Factura_PK
    FROM Cobro JOIN Cobro_Mensual
    ON Cobro.ID_Factura_PK = Cobro_Mensual.ID_Factura_FK)
        SET @Ganancias =    (SELECT COALESCE(SUM(Cobro_Mensual.Monto_Final), 0)
                             FROM Cobro_Mensual  JOIN Cobro ON Cobro.ID_Factura_PK = Cobro_Mensual.ID_Factura_FK
                                                 JOIN Atleta ON Atleta.Codigo_Atleta_PK = Cobro.Codigo_Atleta_FK
        WHERE MONTH(Cobro.Fecha_Pago) = @Mes

        );

    IF EXISTS (
        SELECT Cobro.ID_Factura_PK
        FROM Cobro JOIN Cobro_Individual
        ON Cobro.ID_Factura_PK = Cobro_Individual.ID_Factura_FK)
            SET @Ganancias +=   (SELECT COALESCE(SUM(Cobro_Individual.Monto_Total), 0)
                                 FROM Cobro_Individual   JOIN Cobro ON Cobro.ID_Factura_PK = Cobro_Individual.ID_Factura_FK
                                                         JOIN Atleta ON Atleta.Codigo_Atleta_PK = Cobro.Codigo_Atleta_FK
        WHERE MONTH(Cobro.Fecha_Pago) = @Mes
        );

        RETURN @Ganancias
END;
```

5.4 Casos de prueba para requerimientos funcionales y no funcionales (restricciones)

Caso de Prueba 1: Rango de horas de bloque de entrenamiento	
Propósito:	Verificar que el número de horas de entrenamiento está dentro del rango de 0 a 24 horas, exclusivo
Requerimiento funcional o no funcional asociado:	Restricción declarativa *check* en la definición de tabla Bloque de Entrenamiento.
Pasos:	1. Se insertan individualmente tuplas con valor de rango de horas -1 , 24, 0, 6, 25
Resultado esperado:	Rechaza las tuplas con valor inválido de rango de horas (-1, 24, 0). Acepta la tupla con valor válido 6
Resultado de la prueba:	PASÓ

Caso de Prueba 2: Integridad referencial en la tabla Atleta	
Propósito:	Eliminar a un Atleta en específico y verificar que que se eliminen las tuplas necesarias.
Requerimiento funcional o no funcional asociado:	Restricción de integridad referencial en la definición de la tabla
Pasos:	1. Elegir un atleta y eliminarlo. 2. Hacer *Selects* a las tablas relacionadas con Atleta.
Resultado esperado:	Cobro queda con un Codigo_Atleta_FK de -1, las tuplas de Prueba_Fisica, Bloque_Entrenamiento y Compuesto se eliminan.
Resultado de la prueba:	PASÓ

Caso de Prueba 3: Disparador en cobros mensuales	
Propósito:	Verifica el funcionamiento del disparador en la tabla Cobro_Mensual que asigna Monto_Final considerando el descuento.
Requerimiento funcional o no funcional asociado:	n/a
Pasos:	1. Se inserta una tupla de cobro mensual con 25% de descuento sobre el costo mensual de $60
Resultado esperado:	*Monto_Final = 45*.
Resultado de la prueba:	PASÓ

Caso de Prueba 4: Función para obtener código de atleta a partir del correo	
Propósito:	Verifica que se obtiene el código del atleta cuando existe el correo y en caso contrario, devuelve -1
Requerimiento funcional o no funcional asociado:	Que exista un correo correspondiente del atleta cuyo código es de interés
Pasos:	1. Se ejecuta la función ingresando un correo no existente y otro que sí
Resultado esperado:	Obtener un valor de -1 al ingresar un correo no válido, y obtener el código del atleta correspondiente al ingresar su correo.
Resultado de la prueba:	PASÓ

Caso de Prueba 5: Consulta facturas por atleta	
Propósito:	Verifica el funcionamiento de la consulta de montos pagados por cada atleta
Requerimiento funcional o no funcional asociado:	n/a
Pasos:	1. Se ejecuta el código de consulta
Resultado esperado:	Desplegar las facturas y pagos realizados por cada atleta
Resultado de la prueba:	PASÓ

Caso de Prueba 6: Número de pruebas físicas por atleta (procedimiento almacenado)	
Propósito:	Devuelve la cantidad de pruebas por cada atleta.
Requerimiento funcional o no funcional asociado:	n/a
Pasos:	1. Ejecutar el procedimiento almacenado.
Resultado esperado:	Devuelve números específicos para cada atleta.
Resultado de la prueba:	PASÓ

Caso de Prueba 7: Duplicado de correos	
Propósito:	Insertar un atleta ya existente.
Requerimiento funcional o no funcional asociado:	Restricción declarativa *unique* en la definición de tabla Atleta.
Pasos:	1. Insertar un atleta existente.
Resultado esperado:	Error a la hora de insertar la tupla.
Resultado de la prueba:	PASÓ

Caso de Prueba 8: Integridad referencial en la tabla Bloque	
Propósito:	Verificar el resultado de la eliminación de una tupla de Bloque.
Requerimiento funcional o no funcional asociado:	Declaración de la restricción.
Pasos:	1. Eliminar una tupla en específio.
Resultado esperado:	Se elimina el Bloque y el compuesto.
Resultado de la prueba:	PASÓ

Caso de Prueba 9: Consulta de los bloques	
Propósito:	Consultar los bloques de un atleta en específico.
Requerimiento funcional o no funcional asociado:	n/a
Pasos:	1. Realizar la consulta específica.
Resultado esperado:	Despliegue de las rutinas de un atleta en específico.
Resultado de la prueba:	PASÓ

5.5 Retos de la implementación

En general, a la hora de de implementar la base de datos, no hubo problemas tan graves, desde luego que por la naturaleza tan exclusiva de la DBMS, hay cosas que no se saben, pero en estos casos bastaba con buscar la documentación en línea para poder resolver el problema. Aún así, hubo dos problemas circunstancias sobresalientes que pasaron.

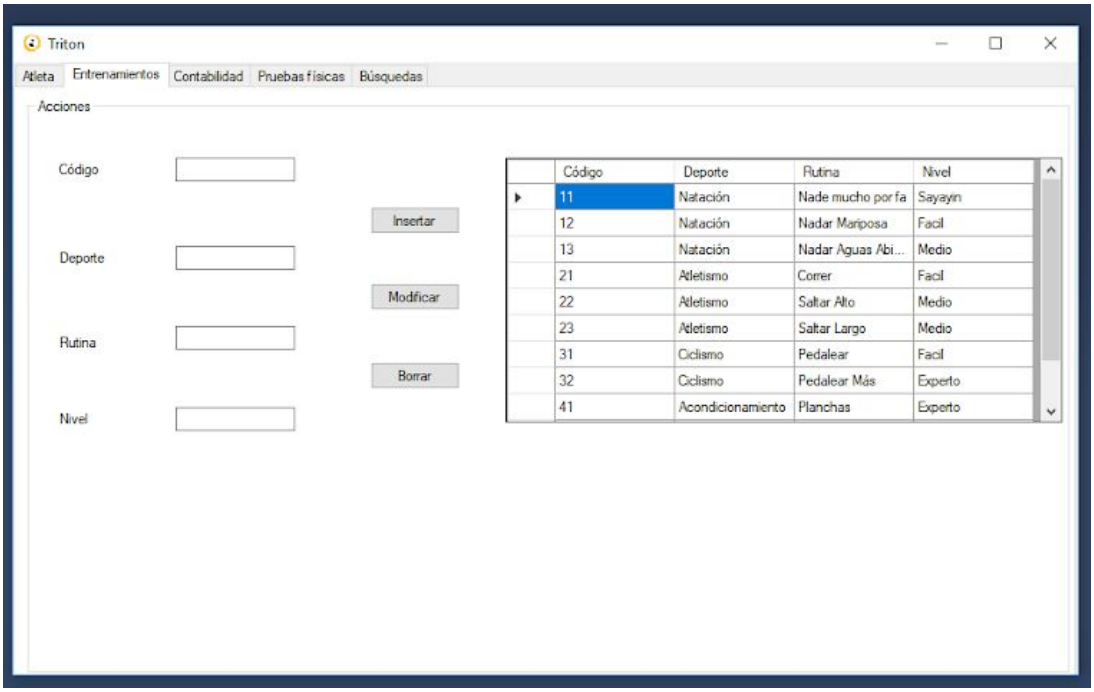
En primer lugar, se tuvo que cambiar la base de datos varias veces, muchas veces a la hora de discutir sobre cómo implementar el diseño, se notaba las falencias de los mismos.

El segundo aspecto fue más grave: la implementación de un disparador que calculara el atributo derivado. Al principio se tenía una propuesta para el disparador que funcionaba correctamente, pero después se vio un problemas que no se había previsto: el disparador se caía a la hora de hacer una sola inserción para varias tuplas. Debido a la DBMS, el disparador se ejecuta una sola vez con el conjunto de tuplas. Luego de varias semanas e intentos con *SELECTS* y demás, el problema se solucionó usando cursores.

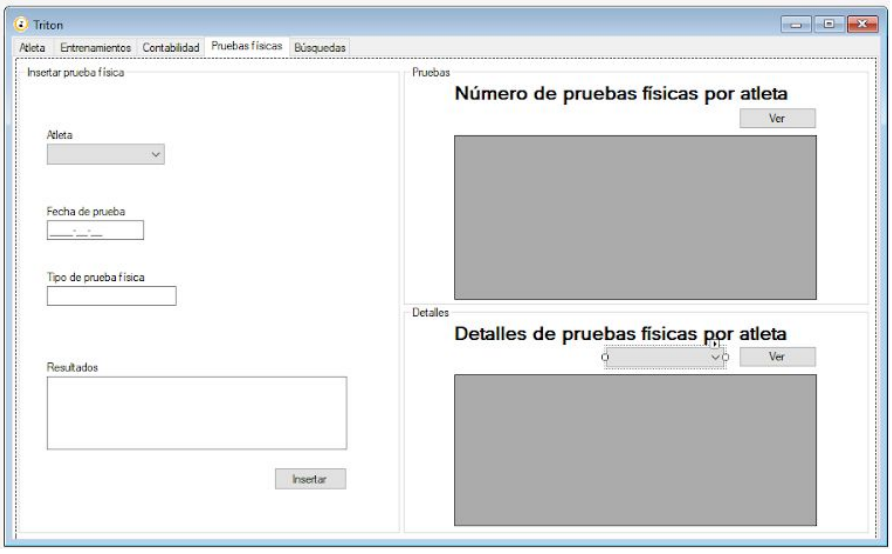
ETAPA 4: Implementación de la interfaz

6 Interfaz: su implementación y pruebas

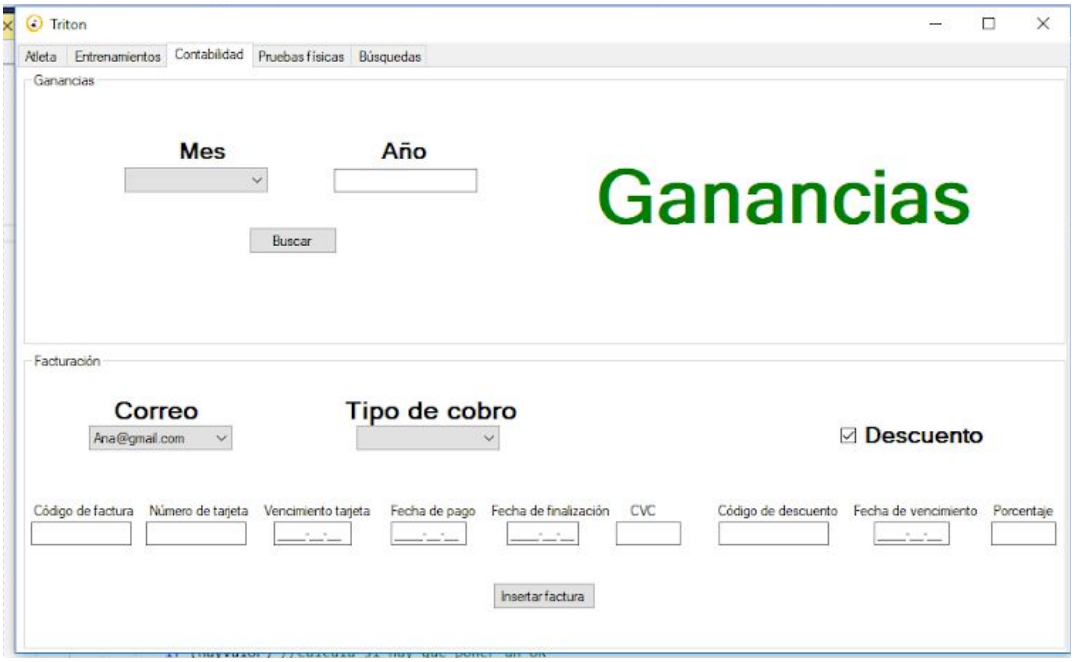
6.1 Algunas vistas de la interfaz



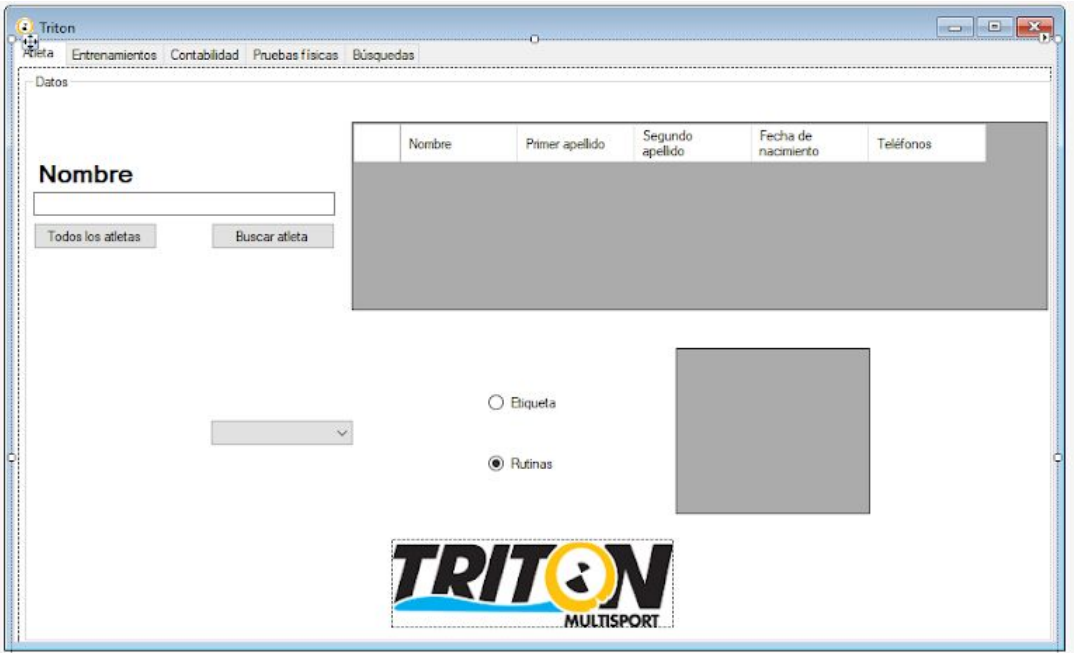
En este caso se pueden insertar, borrar y modificar los entrenamientos individuales. Para insertar es necesario ingresar todos los valores, para modificar solo la llave y el valor a insertar y para borrar es necesario la llave más valores extras.



Para la inserción de pruebas físicas se mete un correo con opciones asistidas, fecha con máscara y el resto de valores. Se pueden ver la cantidad de pruebas físicas o los detalles de él.

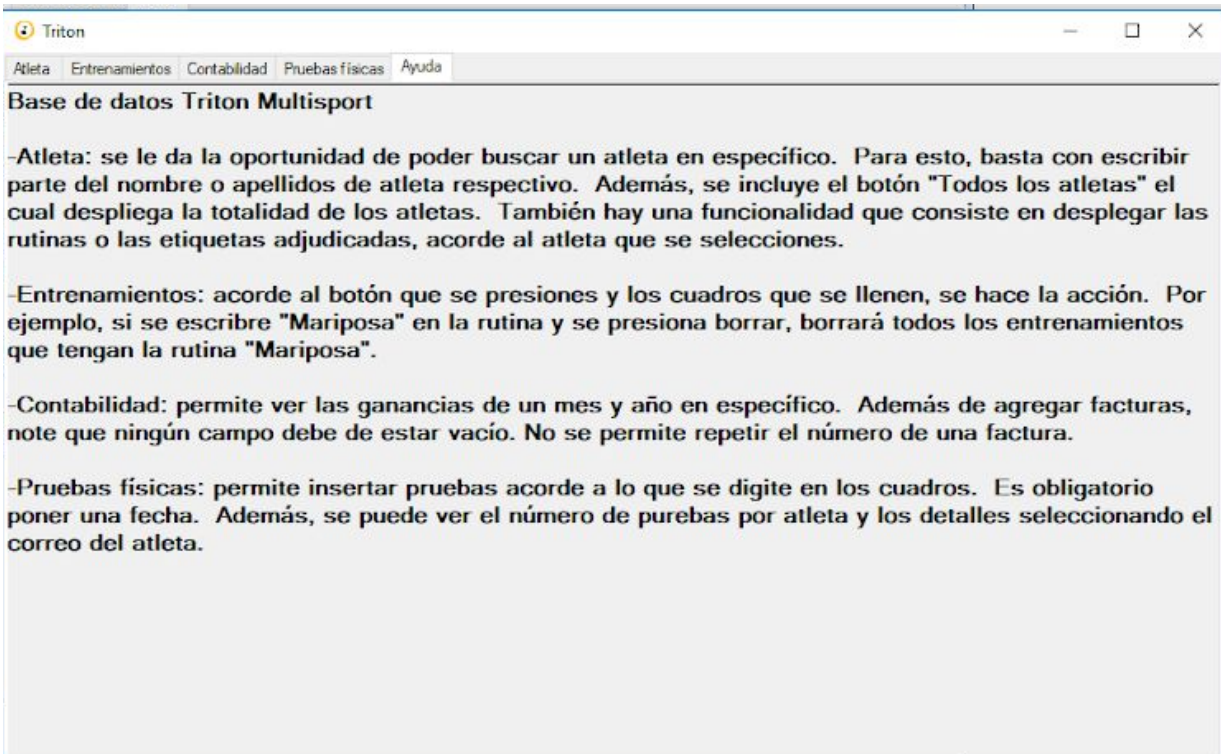


En esta pantalla algunas opciones se activan para algunos valores de los checkbox o de los combo box. Ganancias se calcula con una función almacenada. Y para el descuento se activa un trigger que le reduce un porcentaje al monto.



En la pantalla de atletas se puede consultar la información del los atletas que se presentan mediante un Data Grid View Table. Para esto se puede filtrar la búsqueda

por el nombre o solicitar ver todos los datos. También se pueden ver los datos o las rutinas con marcar unos radio box.



En la última pantalla se muestra las instrucciones de uno de todas las clases del programa.

6.2 Casos de prueba

<Debe documentar al menos casos de pruebas especificados en el enunciado del proyecto referentes a integridad referencial, disparadores, consultas (una para cada diferente número de tablas involucradas) e inyección SQL. Se recomienda utilizar el formato mostrado en la figura 9 para cada caso de prueba.>

Caso de Prueba #: Borrado de Entrenamiento Individual	
Propósito:	Verificar que un borrado de tupla en la tabla Entrenamiento Individual ejecuta un borrado en la tabla hijo Compuesto
Requerimiento funcional o no funcional asociado:	Restricción de llave foránea
Pasos:	1. Eliminar el primer entrenamiento de natación

Resultado esperado:	Se elimina el entrenamiento de natación en la tabla Compuesto
Resultado de la prueba:	PASÓ

Caso de Prueba #: Disparador de descuento	
Propósito:	Verificar el funcionamiento del disparador que aplica al monto final de una factura el descuento, si existe
Requerimiento funcional o no funcional asociado:	n/a
Pasos:	1. Se inserta un nuevo cobro mensual con 50% de descuento sobre el monto mensual de 60$
Resultado esperado:	Se inserta la factura indicando el monto final de 30$ después de aplicar el descuento
Resultado de la prueba:	PASÓ

Caso de Prueba #: Consulta de una tabla: Atleta	
Propósito:	Verificar que se ejecuta exitosamente la consulta de atletas en el sistema
Requerimiento funcional o no funcional asociado:	n/a
Pasos:	1. Se consulta por el atleta Gabriela
Resultado esperado:	Desplegar los datos del atleta
Resultado de la prueba:	PASÓ

Caso de Prueba #: Consulta de dos tablas: Pruebas Físicas	
Propósito:	Verificar la consulta exitosa de pruebas físicas por atleta, haciendo join de ambas tablas
Requerimiento funcional o no funcional asociado:	Llave foránea
Pasos:	1. Se consulta las pruebas del atleta daniel, insertando su correo como identificador
Resultado esperado:	Desplegar los datos de las pruebas físicas del atleta
Resultado de la prueba:	PASÓ

Caso de Prueba #: Consulta de tres tablas: Cobro Mensual	
Propósito:	Consultar la información de los cobros mensuales en el sistema
Requerimiento funcional o no funcional asociado:	Llave foránea
Pasos:	1. Se consulta la información de todos los cobros mensuales, incluyendo datos del atleta
Resultado esperado:	Desplegar los datos de la factura
Resultado de la prueba:	PASÓ

Figura 9. Ejemplo de formato para casos de prueba.

7 Referencias

<Incluir todas las fuentes bibliográficas usando el siguiente formato:

[1] B.Wagner, M.Wenzel, M.Hoffman,, (2018) C# Guide, Recuperado de https://docs.microsoft.com/en-us/dotnet/csharp.

[2] Malinowki.E, Martinez.A. Material de apoyo Base de datos 1. Version 1,(2018).

[3] Entrevista a Brenes.B, (2018). San Pedro, Montes de Oca.