Board Game Round

Todo lo que necesitas saber sobre juegos de mesa

Ingeniería de Software | grupo 2

Javier Fernández Castillo

Javier García Aguilar

Javier Ramírez Núñez

ÍNDICE

1.- INTRODUCCIÓN AL PROBLEMA

El incremento en la divulgación sobre juegos de mesa y el gran interés actual que hay en ellos ha hecho que este mercado aumente de forma considerable. Por ello, a la empresa Zacatrus le interesa dicho crecimiento de este mercado y quiere implementar un software que ayude a los usuarios a obtener información más fácilmente sobre los distintos juegos.

Entre todos los intereses que nos muestra la empresa, encontramos como carácter general la necesidad de interacción del usuario con el sistema, pudiendo puntuar los juegos, opinar, dejar comentarios y guías. Esto aumentará la interacción entre usuarios y ayudará al crecimiento de la comunidad.

Otra función importante que será implementada es la posibilidad de conseguir puntos haciendo ciertas acciones, como opinar, puntuar o comentar; que solo se podrán realizar como usuario registrado. Este sistema de recompensas atraerá de forma considerable a más usuarios, y repercutirá notablemente en la decisión de registrarse. También consideran necesario un gestor de stock para el gerente de la tienda.

Por último, la empresa nos pide un sistema que genere el top juegos por puntuación, juegos más vendidos y distintos campos a considerar.

Una vez recogida la propuesta de la empresa, tenemos como objetivo desarrollar una aplicación que resuelva todos los problemas que se nos plantean e implementar las funcionalidades requeridas.

2.- GLOSARIO DE TÉRMINOS

TÉRMINO	DESCRIPCIÓN
Juego de mesa	Juego que consta de un tablero y fichas de diferentes formas y colores, lo que obliga a que se organice sobre una superficie plana.
Valoración	Puntuación que se le da a un juego del 0 al 10.
Guía	Documentación escrita por un usuario que ayuda a los demás usuarios a entender el juego o mostrar estrategias.
Reseña	Comentario de corta extensión que se hace acerca de lo que te ha parecido un juego.
Expansión	Extensión de un juego que amplía la experiencia del mismo y que suele dar diversas mecánicas nuevas al juego principal.
Recompensas	Son los premios obtenidos al acumular cierta cantidad de puntos.
Puntos	Se obtienen realizando diversas acciones como comentar, opinar o puntuar, y sirven para obtener recompensas.
Dificultad	Nivel de asequibilidad que nos plantea el juego para poder jugarlo.
Foro	Lugar donde poder interactuar con la comunidad sobre cualquier tema relacionado con los juegos.
Comunidad	Conjunto de personas que acceden habitualmente a la página y que tienen un gusto común, en este caso, por los juegos de mesa.
Comentarios	Mensajes de opinión que hacen los usuarios sobre los distintos juegos, con la posibilidad de ser respondido por otros usuarios.

3.- VISIÓN GENERAL DEL SISTEMA

3.1.- REQUISITOS GENERALES

OBJ-001 Venta de Juegos
Como empresa**.**
Quiero aumentar el número de ventas que tiene nuestra tienda.
Para incrementar los beneficios de la empresa.

OBJ-002 Gestión de tienda
Como gerente de la tienda.
Quiero tener un control acerca del stock de juegos en la tienda.
Para obtener información de la demanda de juegos de mesa.

OBJ-003 Gestión de puntos
Como usuario.
Quiero un sistema de puntos por los cuales recibamos descuentos.
Para aumentar el interés de la comunidad.

OBJ-004 Divulgar información sobre juegos de mesa
Como director de marketing.
Quiero aumentar la información al alcance del público acerca de los juegos de mesa.
Para atraer un mayor número de usuarios.

3.2.- USUARIOS DEL SISTEMA

Director

Principal encargado de dirigir la empresa y opinar acerca del proyecto.

Administrador

Persona encargada del foro, censura, y algunos detalles de la página web.

Usuario

Persona registrada o no que utiliza la página para obtener información acerca de juegos de mesa.

Gerente de la tienda

Se encarga de llevar la tienda y realizar los pedidos al almacén.

Director de marketing

Se encarga de publicitar y facilitar información para atraer a compradores.

Director comercial

Se encarga de dirigir el stock y administrar comercialmente la empresa.

4.- CATÁLOGO DE REQUISITOS

4.1.- REQUISITOS DE INFORMACIÓN

RI-001 Información sobre un producto
Como director comercial.
Quiero disponer de la información correspondiente a los detalles del producto. Para cada producto necesitamos conocer nombre del juego, precio, categoría del juego, código EAN (DNI de cada producto), fecha de última venta, ratio de depreciación (precio/(número de días desde última venta/90)) y stock.
Para facilitar la distribución de los juegos.

RI-002 Información sobre la venta de un producto
Como director comercial.
Quiero disponer de la información correspondiente a la venta de un juego. Para cada juego de mesa necesitamos conocer precio, imágenes, fecha de la venta realizada, dirección de envío y usuario al que se le envía.
Para

RI-003 Información sobre las ventas de las tiendas
Como director comercial.
Quiero disponer de la información correspondiente al a las ventas de las tiendas. Para el conjunto de venta quiero conocer el nombre de cada juego, su precio, número de ventas, fecha de última venta y dinero recaudado total.
Para dar a conocer los beneficios que nos aporta cada juego.

RI-004 Información sobre usuario
Como
Quiero disponer de la información correspondiente al usuario. Para cada usuario necesitamos su nombre, apellido, nombre de usuario, DNI, correo electrónico, dirección, edad, password
Para

RI-005 Información sobre puntos

Como usuario.

Quiero disponer de puntos para conseguir descuentos en los juegos. Para cada usuario tendremos una cantidad de puntos acumulados, que se puede obtener de varias formas. Opinando, comentando o puntuando conseguirás un punto. Por cada euro gastado obtendrás 3 puntos. Para ello requerimos de nombre de usuario, número de comentarios/opiniones/puntuaciones dadas y euros gastados en los juegos.

Para tener información sobre los puntos que tenemos.

RI-006 Información sobre juegos de mesa

Como usuario.

Quiero disponer de la información correspondiente a los juegos de mesa. Para cada juego de mesa necesitamos conocer título, autor, categoría, edad recomendada, número de jugadores, tiempo de juegos, editorial, contenido, estado en tienda, puntuación del juego, fecha de lanzamiento y fotos.

Para dar a conocer información sobre los distintos juegos.

RI-007 Información sobre comentarios y puntuaciones

Como usuario.

Quiero disponer de interacciones que me aporten información o me dejen aportar mi sensaciones sobre un juego. Para cada juego podremos comentar o puntuar. Para ello disponemos del usuario que interactúa, juego con el que decide interactuar y el comentario o puntuación que decide poner.

Para tener información sobre los puntos que tenemos.

4.2.- REGLAS DE NEGOCIO

RN-001 Stock bajo
Como director comercial.
Quiero que se cumpla la siguiente regla de negocio: cuando el stock de un producto sea bajo (menor que 5) se le notificará al usuario de ello.
Para incrementar el interés del usuario por la compra del producto

RN-002 Precio mínimo
Como director comercial.
Quiero que se cumpla la siguiente regla de negocio: el descuento de un producto no pueda ser superior al 50% del valor del producto.
Para no entrar en quiebra.

RN-003 Usuario Registrado
Como administrador.
Quiero que se cumpla la siguiente regla de negocio: el nombre de usuario no puede tener más de 15 caracteres y no puede haber 2 usuarios con el mismo nombre.
Para que no haya nombres excesivamente largos.

RN-004 Edad mínima
Como director comercial.
Quiero que se cumpla la siguiente regla de negocio: no puede registrarse si su edad es inferior a 13 años.
Para

RN-005 Cuántos puntos otorga cada interacción

Como usuario.

Quiero que se cumpla la siguiente regla de negocio: los puntos se otorguen de varias formas. Un usuario puede obtener un punto por cada comentario o valoración que se haga de un juego. O 3 puntos por cada euro gastado. Cada punto equivale a un céntimo.

Para tener conocimiento de cómo funciona la obtención de puntos

RN-006 Puntos no pueden ser negativos

Como director comercial.

Quiero que se cumpla la siguiente regla de negocio: los puntos no pueden ser negativos.

Para que no haya fallos en la base de datos.

RN-007 Revisión de comentarios por el moderador

Como moderador.

Quiero que se cumpla la siguiente regla de negocio: los comentarios que no tengan sentido o no aporten nada sean eliminados. Reduciendo el punto por comentar u opinar que ha obtenido el usuario.

Para dar la opción a los usuarios a conservar sus puntos si quieren acumularlo y utilizar más en un futuro.

RN-008 Número de carácteres de un comentario

Como administrador.

Quiero que se cumpla la siguiente regla de negocio: el número de caracteres de un comentario no pueda superar los 500 caracteres.

Para no tener comentarios demasiado extensos y aburridos de leer.

4.3.- REQUISITOS FUNCIONALES

RF-001 Ranking de juegos vendidos
Como gerente de tienda.
Quiero conocer los juegos más vendidos en forma de ranking.
Para orientarse a la hora de recomponer el stock de la tienda.

RF-002 Ranking de juegos de mesa.
Como usuario.
Quiero conocer los juegos con mayor puntuación.
Para orientarse a la hora de comprar.

RF-003 Ventas de un juego en un periodo de tiempo
Como director comercial.
Quiero consultar cuantas ventas se han realizado de un juego en un periodo de tiempo.
Para

RF-004 Ingresos en un periodo de tiempo
Como director comercial.
Quiero saber cuantos ingresos he obtenido en un periodo de tiempo
Para

RF-005 Juegos vendidos en un día
Como gerente de tienda.
Quiero saber el listado de juegos que se han vendido en un día.
Para poder tener información de los juegos que más se venden

RF-006 Consultar el stock de cada producto
Como gerente de tienda.
Quiero consultar el stock de un producto.
Para poder gestionar correctamente mi tienda.

RF-007 Reposición de stock
Como gerente de tienda.
Quiero realizar pedidos de juegos de mesa y registrar los productos cuando los reciba.
Para garantizar que hay siempre suficientes productos de venta.

RF-008 Juego en stock
Como gerente de tienda.
Quiero registrar que un juego que se ha agotado vuelve a estar operativo.
Para notificar a los usuarios que de nuevo hay disponibilidad de un juego.

RF-009 Listado de pedidos por usuarios
Como director comercial.
Quiero que aparezcan listas de todos los pedidos realizados por cada usuario
Para personalizar el tipo de recomendaciones que se le hace a cada usuario

RF-010 Clientes más frecuentes
Como administrador.
Quiero saber qué clientes son más habituales.
Para conocer quiénes visitan más la página y realiza más compras para tenerlo más en consideración para beneficiarlo.

RF-011 Sistema de puntos
Como usuario.
Quiero que cada vez que se realice una interacción o la compra de un juego se acumulen los puntos correspondientes. Siendo la fórmula: Puntos obtenidos= Nº de comentarios/puntuaciones + (Dinero gastado en compras * 3)
Para obtener descuentos en mis proximas compras.

5.- PRUEBAS DE ACEPTACIÓN

5.1.- PRUEBA DE ACEPTACIÓN DE UNA REGLA DE NEGOCIO

RN-001 Stock bajo

Como director comercial.

Quiero que se cumpla la siguiente regla de negocio: cuando el stock de un producto sea bajo se le notificará al usuario de ello.

Para incrementar el interés del usuario por la compra del producto

Prueba de aceptación de stock bajo

- El aviso no se recibe si el stock es superior a 5.
- El juego notifica que tiene stock bajo.

RN-002 Precio mínimo

Como director comercial.

Quiero que se cumpla la siguiente regla de negocio: el descuento de un producto no pueda ser superior al 50% del valor del producto.

Para no entrar en quiebra.

Prueba de aceptación de precio mínimo

- Si se intenta poner un descuento superior a un 50% el sistema lo impide.
- Si el descuento es inferior a un 50% se aplica correctamente.

RN-003 Usuario Registrado

Como administrador.

Quiero que se cumpla la siguiente regla de negocio: el nombre de usuario no puede tener más de 15 caracteres y no puede haber 2 usuarios con el mismo nombre.

Para que no haya nombres excesivamente largos.

Prueba de aceptación de usuario registrado

- Un usuario intenta ponerse un nombre con más de 15 caracteres, se le avisa de que es demasiado largo y debe cambiarlo
- Un usuario intenta ponerse un nombre que ya existe se le avisa de que es ya existe y debe cambiarlo
- Un usuario intenta ponerse un nombre con menos de 15 caracteres, y si nadie más tiene ese mismo, el nombre es válido.

RN-004 Edad mínima

Como director comercial.

Quiero que se cumpla la siguiente regla de negocio: no puede registrarse si su edad es inferior a 13 años.

Para

Prueba de aceptación de edad mínima

- El usuario introduce una edad inferior a 13 años, se le impide registrarse en la web.
- El usuario introduce una edad superior a 13 años, puede seguir registrándose sin problemas.

RN-005 Cuántos puntos otorga cada interacción

Como usuario.

Quiero que los puntos se otorguen de varias formas. Un usuario puede obtener un punto por cada comentario o valoración que se haga de un juego. O 3 puntos por cada euro gastado. Cada punto equivale a un céntimo.

Para tener conocimiento de cómo funciona la obtención de puntos

Prueba de aceptación de cuántos puntos otorga cada interacción

- Un usuario realiza un comentario o da una valoración, entonces se le añade un punto a los puntos actuales.
- Un usuario realiza una compra, entonces se le añade a sus puntos actuales el número de puntos correspondiente a dicha compra (3*Importe de la compra)

RN-006 Puntos no pueden ser negativos

Como director comercial.

Quiero que los puntos no puedan ser negativos.

Para que no haya fallos en la base de datos.

Prueba de aceptación de puntos no pueden ser negativos

- Si los puntos correspondientes a un usuario son positivos, todo va bien.
- Si los puntos de un usuario son negativos, salta un error que dice que te pongas en contacto con el soporte técnico.

RN-007 Revisión de comentarios por el moderador

Como moderador.

Quiero que los comentarios que no tengan sentido o no aporten nada sean eliminados. Reduciendo el punto por comentar u opinar que ha obtenido el usuario.

Para dar la opción a los usuarios a conservar sus puntos si quieren acumularlo y utilizar más en un futuro.

Prueba de aceptación de revisión de comentarios por el moderador

- Un moderador ve un comentario poco apropiado entonces decide eliminarlo y su usuario pierde el punto correspondiente a ese comentario.

RN-008 Número de carácteres de un comentario

Como administrador.

Quiero que se cumpla la siguiente regla de negocio: el número de caracteres de un comentario no pueda superar los 500 caracteres.

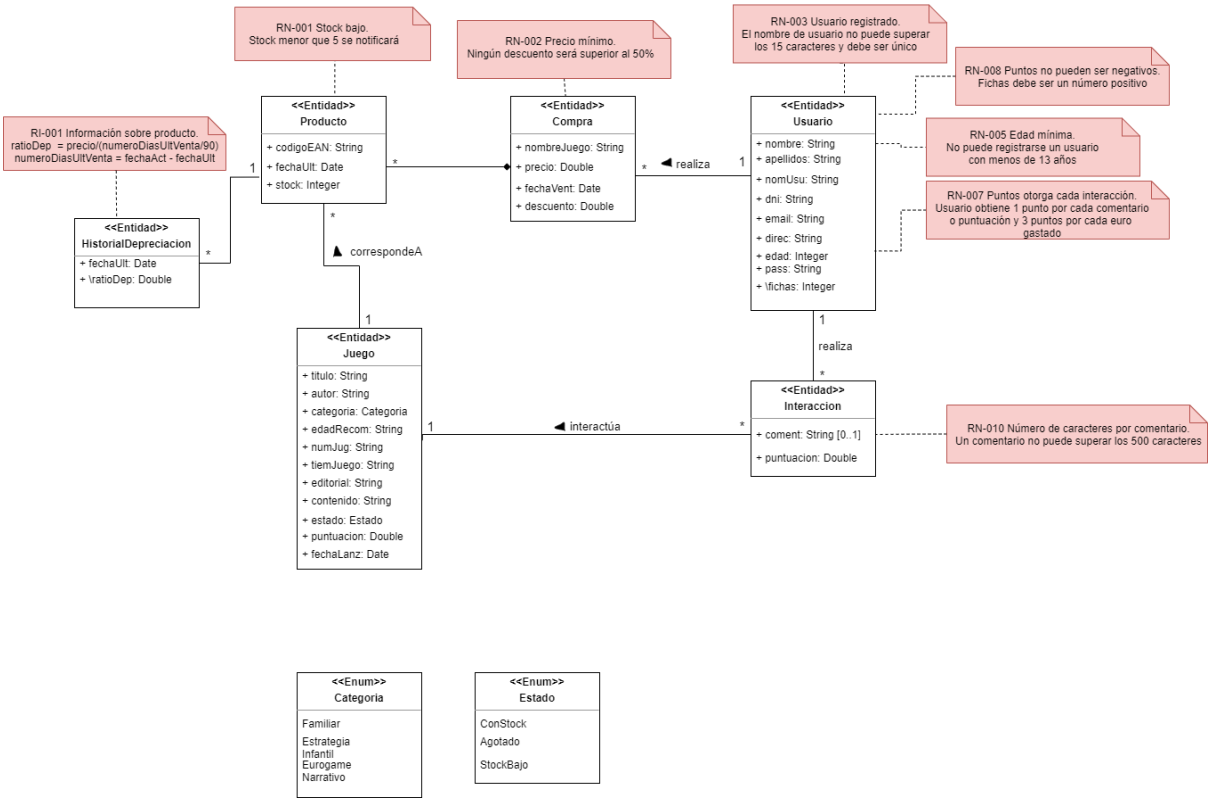
Para no tener comentarios demasiado extensos y aburridos de leer.

Prueba de aceptación de número de carácteres de un comentario/opinión

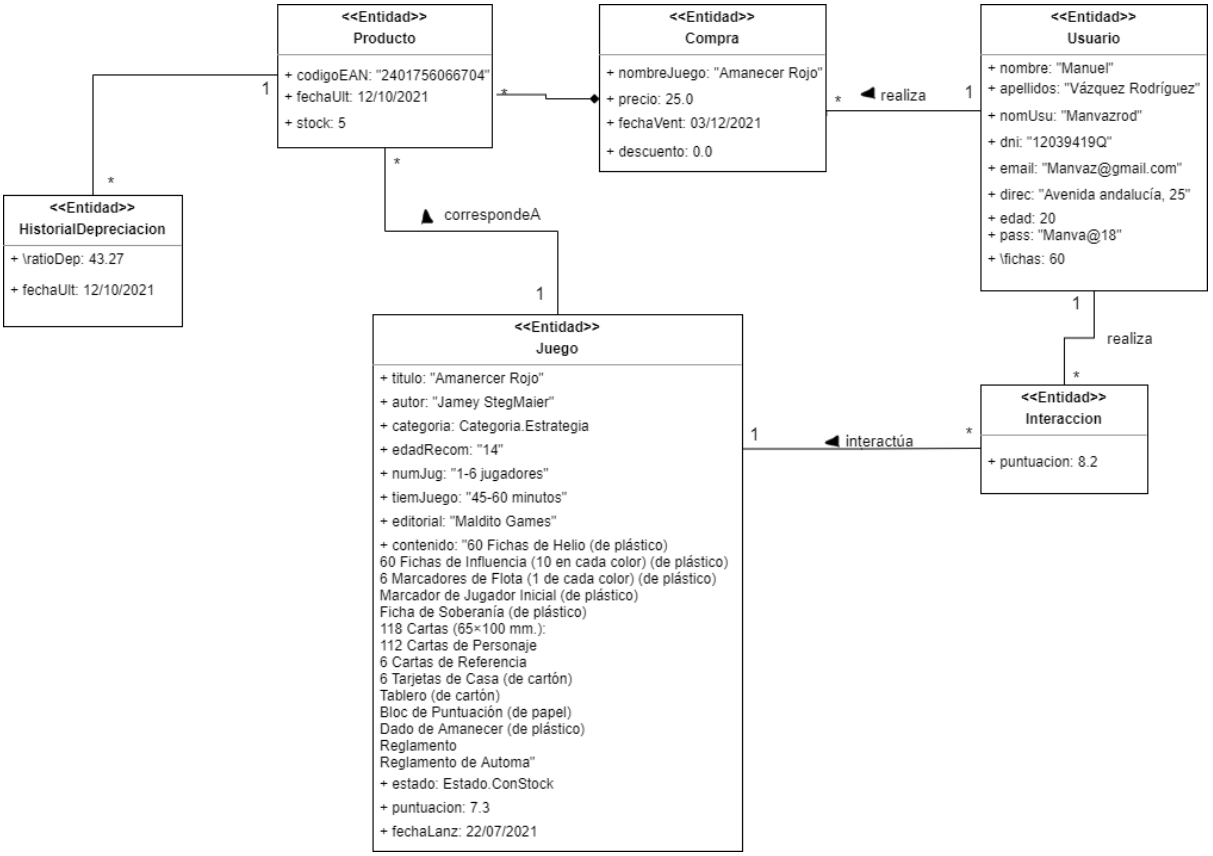
- Un comentario tiene más de 500 caracteres, entonces se recibe un mensaje de error
- Un comentario tiene menos de 500 caracteres y se publica correctamente.

6.- MODELO CONCEPTUAL

6.1.- DIAGRAMAS DE CLASES UML CON RESTRICCIONES



6.2.- ESCENARIOS DE PRUEBA



7.- MATRICES DE TRAZABILIDAD

	RN-001 Stock bajo	RN-002 Precio mínimo	RN-003 Usuario Registrado	RN-004 Edad mínima
RI-001 Información sobre productos	X	X		
RI-002 Información sobre la venta de un producto		X		
RI-003 Información sobre las ventas de las tiendas				
RI-004 Información sobre usuario			X	X
RI-005 Información sobre puntos				
RI-006 Información sobre juegos de mesa				
RI-007 Información sobre comentarios y puntuaciones				

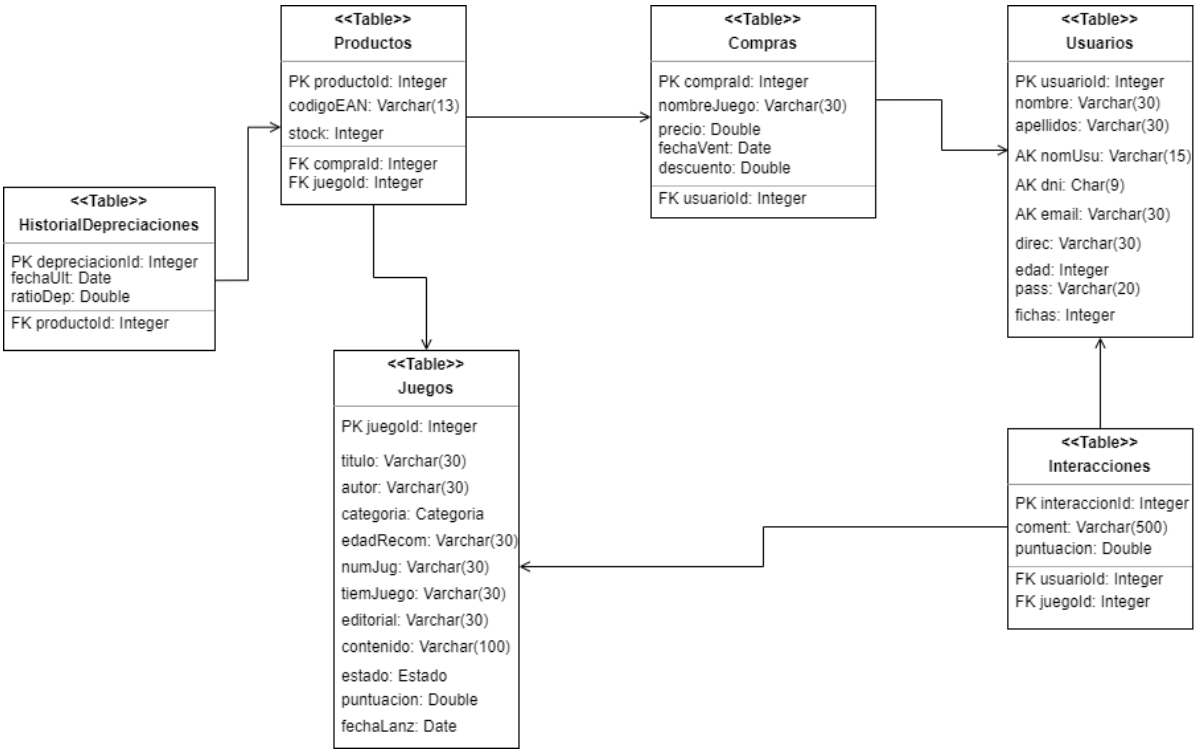
	RN-005 Cuántos puntos otorga cada interacción	RN-006 Puntos no pueden ser negativos	RN-007 Revisión de comentarios por el moderador	RN-008 Número de carácteres de un comentario
RI-001 Información sobre productos				
RI-002 Información sobre la venta de un producto				
RI-003 Información sobre las ventas de las tiendas				
RI-004 Información sobre usuario				
RI-005 Información sobre puntos	X	X		
RI-006 Información sobre juegos de mesa				
RI-007 Información sobre comentarios y puntuaciones			X	X

	RF-001 Ranking de juegos vendidos	RF-002 Ranking de juegos de mesa.	RF-003 Ventas de un juego en un periodo de tiempo	RF-004 Ingresos en un periodo de tiempo	RF-005 Juegos vendidos en un día
RI-001 Información sobre productos	X		X	X	X
RI-002 Información sobre la venta de un producto	X		X	X	X
RI-003 Información sobre las ventas de las tiendas	X		X	X	X
RI-004 Información sobre usuario					
RI-005 Información sobre puntos					
RI-006 Información sobre juegos de mesa		X			
RI-007 Información sobre comentarios y puntuaciones		X			

	RF-006 Consultar el stock de cada producto	RF-007 Reposición de stock	RF-008 Juego en stock	RF-009 Listado de pedidos por usuarios	RF-010 Clientes más frecuentes
RI-001 Información sobre productos	X	X	X		
RI-002 Información sobre la venta de un producto	X	X	X		
RI-003 Información sobre las ventas de las tiendas	X	X	X	X	
RI-004 Información sobre usuario				X	X
RI-005 Información sobre puntos					
RI-006 Información sobre juegos de mesa					
RI-007 Información sobre comentarios y puntuaciones					

	RF-011 Sistema de puntos
RI-001 Información sobre productos	
RI-002 Información sobre la venta de un producto	
RI-003 Información sobre las ventas de las tiendas	
RI-004 Información sobre usuario	X
RI-005 Información sobre puntos	X
RI-006 Información sobre juegos de mesa	
RI-007 Información sobre comentarios y puntuaciones	X

8.- MODELO RELACIONAL EN 3FN



8.1.- JUSTIFICACIÓN MODELO RELACIONAL

Para pasar del modelo conceptual al modelo relacional se han seguido los siguientes pasos.

Transformación de la entidad:

Su nombre se pasa a plural y se mantienen los atributos.
Añadimos un Id a cada clave primaria.

Transformación de asociaciones:

Las asociaciones eran del tipo 1:n que se representan con una clave ajena en el rol n.

Finalmente se han puesto en 3FN, en la cual está ya que a ningún atributo se le asigna más de un valor, los atributos no primos dependen de la clave candidata y ningún atributo no primo depende transitivamente de ninguna clave candidata.

9.- MODELO TECNOLÓGICO

1- Creación de base de datos

```
CREATE DATABASE Board_Game_Round;
```

2- Creación de tablas

Juegos

```
CREATE TABLE juegos (
        juegoId INT NOT NULL AUTO_INCREMENT,
        titulo VARCHAR(30) NOT NULL,
        autor VARCHAR(30) NOT NULL,
        categoria ENUM('Familiar','Estrategia','Infantil','Eurogame','Narrativo'),
        numJug VARCHAR(30) NOT NULL,
        tiemJuego VARCHAR(30) NOT NULL,
        editorial VARCHAR(30) NOT NULL,
        contenido VARCHAR(100),
        estado ENUM('conStock','agotado','stockBajos') NOT NULL,
        puntuacion DOUBLE DEFAULT 0.0,
        fechaLanz DATE NOT NULL,
        PRIMARY KEY (juegoId)
);
```

Usuarios

```
CREATE TABLE usuarios(
        usuarioId INT NOT NULL AUTO_INCREMENT,
        nombre VARCHAR(30) NOT NULL,
        apellido VARCHAR(70) NOT NULL,
        nomUsu VARCHAR(15) NOT NULL,
        dni CHAR(9) NOT NULL,
        email VARCHAR(64) NOT NULL,
        direccion VARCHAR(50) NOT NULL,
        edad INT NOT NULL,
        pass VARCHAR(20) NOT NULL,
        fichas INT DEFAULT 0,
        PRIMARY KEY (usuarioId),
        UNIQUE (nomUsu, dni, email),
        CONSTRAINT limite_edad CHECK (edad > 13),
        CONSTRAINT limite_fichas CHECK (fichas >= 0)
);
```

Interacciones

```
CREATE TABLE Interacciones(
        interaccionesId INT NOT NULL AUTO_INCREMENT,
        coment VARCHAR(500),
        puntuación DOUBLE NOT NULL,
        usuarioId INT NOT NULL,
        juegoId INT NOT NULL,
        PRIMARY KEY (interaccionesId),
        CONSTRAINT fk_id_usuario FOREIGN KEY (usuarioId) REFERENCES
        board_game_round.usuarios (usuarioId)
        ON UPDATE RESTRICT ON DELETE RESTRICT,
        CONSTRAINT fk_id_juego FOREIGN KEY (juegoId) REFERENCES
        board_game_round.juegos (juegoId)
        ON UPDATE RESTRICT ON DELETE RESTRICT,
        CONSTRAINT limite_comentario CHECK (CHAR_LENGTH(coment) <= 500),
        CONSTRAINT intervalo_puntuación CHECK
         (puntuación >= 0 AND puntuación <= 5)
);
```

Compras

```
CREATE TABLE compras(
        compraId INT NOT NULL AUTO_INCREMENT,
        nombreJuego VARCHAR(30) NOT NULL,
        precio DOUBLE NOT NULL,
        fechaVent DATE NOT NULL DEFAULT NOW(),
        descuento DOUBLE DEFAULT NULL,
        usuarioId INT NOT NULL,
        PRIMARY KEY(compraId),
        CONSTRAINT fk_id_usuario2 FOREIGN KEY (usuarioId) REFERENCES
        board_game_round.usuarios (usuarioId)
        ON UPDATE RESTRICT ON DELETE RESTRICT,
        CONSTRAINT descuento_max CHECK  (0.5 > descuento >= 0)
);
```

Productos

```
CREATE TABLE productos(
        productoId INT NOT NULL AUTO_INCREMENT,
        codigoEAN VARCHAR(13) NOT NULL,
        stock INT DEFAULT 0,
        compraId INT NOT NULL,
        juegoId INT NOT NULL,
        PRIMARY KEY(productoId),
        CONSTRAINT fk_id_compra FOREIGN KEY (compraId) REFERENCES
        board_game_round.compras (compraId)
        ON UPDATE RESTRICT ON DELETE RESTRICT,
        CONSTRAINT fk_id_juego2 FOREIGN KEY (juegoId) REFERENCES
        board_game_round.juegos (juegoId)
        ON UPDATE RESTRICT ON DELETE RESTRICT,
        CONSTRAINT códigoEAN_correcto CHECK  (CHAR_LENGTH(codigoEAN) = 13),
        CONSTRAINT stock_positivo CHECK  (stock>=0)
);
```

Historial de depreciación

```
CREATE TABLE historial_depreciaciones(
        depreciacionId INT NOT NULL AUTO_INCREMENT,
        fechaUlt DATE DEFAULT NULL,
        ratioDep DOUBLE DEFAULT NULL,
        productoId INT NOT NULL,
        PRIMARY KEY(depreciacionId),
        CONSTRAINT fk_id_producto FOREIGN KEY (productoId) REFERENCES
        board_game_round.productos (productoId)
        ON UPDATE RESTRICT ON DELETE RESTRICT
);
```

2.1- Inserto de datos (para comprobar que todo funciona)

```
INSERT INTO juegos(juegoId, titulo, autor, categoria, numJug, tiemJuego, editorial,
contenido, estado, puntuacion, fechaLanz) VALUES
        (1, 'Oca', 'Maurice Ravel', 'Familiar', '4 jugadores', '5-20 minutos', 'aldearac',
         'dados, cubos, fichas', 'conStock', 3.3, '2020-09-10'),
        (2, 'Monopoly', 'Elizabeth Magie', 'familiar', '8 jugadores', '60 minutos', 'Hasbro',
        NULL, 'stockBajos', 4.5, '1935-11-05'),
        (3, 'Cluedo', 'Lledo', 'Estrategia', '2-6 jugadores', '120 minutos', 'Hasbro', NULL,
         'conStock', 2.5, '1948-08-07');


INSERT INTO usuarios (usuarioId, nombre, apellido, nomUsu, dni, email, direccion,
edad, pass, fichas) VALUES
        (1, 'Manu', 'Fernández', 'ManFer','24918420P','ManFer@gmail.com','Av
        Andalucia', 21, 'puertas90', 240),
        (2, 'Rafael', 'González', 'RaGon','31556098I','RaGon@gmail.com','Av
        Escapulario', 45, 'mansui89', 10),
        (3, 'Javier', 'García', 'JaGar','93791009R','JaGar@gmail.com','Av Luis Montoto',
        20, 'genopl13', 100);


INSERT INTO compras(compraId, nombreJuego, precio, fechaVent, descuento,
usuarioId) VALUES
        (1, 'Monopoly', 20.0, '2021-10-09', 0, 1),
        (2, 'Cluedo', 40.5, '2021-11-05', 0, 1),
        (3, 'Oca', 10.0, '2021-11-28', 0, 3),
        (4, 'Cluedo', 40.5, '2021-12-12', 0, 2);


INSERT INTO interacciones (interaccionesId, coment, puntuación, usuarioId, juegoId)
VALUES
        (1, 'Estuvo chulo', 3.4, 1, 1),
        (2, 'Estuvo guay', 4.5, 3, 2),
        (3, 'Estuvo espectacular', 2.1, 2, 3);


INSERT INTO productos (productoId, codigoEAN, stock, compraId, juegoId) VALUES
        (1, '8491029481023', 18, 1, 2),
        (2, '3415634298181', 4, 3, 1),
        (3, '0491021281023', 10, 2, 3),
        (4, '0491021281023', 9, 4, 3);
```

3- Funciones, procedimientos y

Ranking mejores ventas (RF-001)

```
DELIMITER //
CREATE OR REPLACE PROCEDURE ranking_ventas()
        BEGIN
                SELECT nombreJuego, COUNT(juegoId) ventas
                FROM productos NATURAL JOIN compras
                GROUP BY juegoId ORDER BY ventas DESC;
        END //
DELIMITER ;
```

Ranking mayor puntuación juegos (RF-002)

```
DELIMITER //
CREATE OR REPLACE PROCEDURE ranking_puntuacion_juegos()
        BEGIN
                SELECT titulo, puntuacion
                FROM juegos ORDER BY puntuacion DESC;
END //
DELIMITER ;
```

Ventas de un juego en un periodo de tiempo (RF-003)

```
DELIMITER //
CREATE OR REPLACE FUNCTION ventas_juego_intervalo (fi DATE, ff DATE, nombre
VARCHAR(80)) RETURNS INTEGER
        BEGIN
                RETURN (SELECT COUNT(nombre) ventas
                FROM compras
                WHERE compras.nombreJuego=nombre AND (compras.fechaVent
BETWEEN fi AND ff));
        END //
DELIMITER ;
```

Ingresos en un intervalo de tiempo (RF-004)

```
DELIMITER //
CREATE OR REPLACE FUNCTION ingresos_intervalo (fi DATE, ff DATE) RETURNS
DOUBLE
        BEGIN
                RETURN (SELECT SUM(precio) ingresos
                FROM compras
                WHERE(compras.fechaVent BETWEEN fi AND ff));
        END //
DELIMITER ;
```

Juegos vendidos en un dia (RF-005)

```
DELIMITER //
CREATE OR REPLACE PROCEDURE  venta_dia2 (d DATE)
        BEGIN
                SELECT nombreJuego, fechaVent, COUNT(nombreJuego) cantidad
                FROM compras
                WHERE(compras.fechaVent = d) GROUP BY nombreJuego;
        END //
DELIMITER ;
```

Stock de un juego (RF-006)

```
DELIMITER //
CREATE OR REPLACE FUNCTION  stock_juego (juego VARCHAR(30)) RETURNS INT
        BEGIN
                RETURN (SELECT MIN(stock) stock
                FROM productos NATURAL JOIN compras
                WHERE(compras.nombreJuego = juego));
        END //
DELIMITER ;
```

Reposición de stock (RF-007)

```
DELIMITER //
CREATE OR REPLACE PROCEDURE  reposicion_stock (juego VARCHAR(30), n INT)
        BEGIN
                UPDATE productos p JOIN compras c
                ON p.compraId = c.compraId
                SET stock = n
                WHERE c.nombreJuego=juego;
        END //
DELIMITER ;
```

Estado de juego (RF-008)(RN-001)

```
DELIMITER //
CREATE OR REPLACE TRIGGER estado_agotado
        AFTER UPDATE ON productos
        FOR EACH ROW
        BEGIN
                UPDATE juegos NATURAL JOIN productos
                SET juegos.estado = 'agotado'
                WHERE productos.stock=0
                AND productos.juegoId=juegos.juegoId;
        END //
DELIMITER ;
```

```
DELIMITER //
CREATE OR REPLACE TRIGGER estado_stockbajo
        AFTER UPDATE ON productos
        FOR EACH ROW
        BEGIN
                UPDATE juegos NATURAL JOIN productos
                SET juegos.estado = 'stockBajos'
                WHERE productos.stock BETWEEN 1 AND 5
                AND productos.juegoId=juegos.juegoId;
        END //
DELIMITER ;
```

```
DELIMITER //
CREATE OR REPLACE TRIGGER estado_constock
        AFTER UPDATE ON productos
        FOR EACH ROW
        BEGIN
                UPDATE juegos NATURAL JOIN productos
                SET juegos.estado = 'conStock'
                WHERE productos.stock>5
                AND productos.juegoId=juegos.juegoId;
        END //
DELIMITER ;
```

Pedidos por usuario (RF-009)

```
DELIMITER //
CREATE OR REPLACE PROCEDURE pedidos_usuario()
        BEGIN
                SELECT nomUsu,  nombreJuego, precio, fechaVent, descuento
                FROM compras c NATURAL JOIN usuarios u
                WHERE c.usuarioId=u.usuarioId;
        END //
DELIMITER ;
```

Clientes más frecuentes (RF-010)

```
DELIMITER //
CREATE OR REPLACE PROCEDURE clientes_mas_frecuentes()
        BEGIN
                SELECT nomUsu,  COUNT(usuarioId) pedidos_realizados
                FROM compras c NATURAL JOIN usuarios u
                GROUP BY nomUsu ORDER BY pedidos_realizados DESC;
        END //
DELIMITER ;
```

Sistema de puntuación (RF-011)

```
DELIMITER //
CREATE OR REPLACE TRIGGER sistema_puntuacion_compras
        AFTER INSERT ON compras
        FOR EACH ROW
        UPDATE usuarios u NATURAL JOIN compras c
        SET u.fichas = (SELECT  (3*(SELECT precio FROM compras ORDER BY
        compraId DESC LIMIT 1)+ (SELECT fichas FROM usuarios NATURAL JOIN
        compras  WHERE (SELECT usuarioId FROM compras ORDER BY compraId DESC
        LIMIT 1)=compras.usuarioId LIMIT 1)) FROM usuarios NATURAL JOIN compras
        GROUP BY usuarioId ORDER BY compraId DESC LIMIT 1)
        WHERE (SELECT usuarioId FROM compras ORDER BY compraId DESC LIMIT
        1)=c.usuarioId //
DELIMITER ;

DELIMITER //
CREATE OR REPLACE TRIGGER sistema_puntuacion_interacciones
AFTER INSERT ON interacciones
FOR EACH ROW
        UPDATE usuarios u NATURAL JOIN interacciones i
        SET u.fichas = (SELECT  (1 + (SELECT fichas FROM usuarios
        NATURAL JOIN interacciones WHERE (SELECT usuarioId FROM interacciones
        ORDER BY interaccionesId DESC LIMIT 1)=interacciones.usuarioId LIMIT 1))
        FROM usuarios NATURAL JOIN interacciones GROUP BY usuarioId ORDER BY
        interaccionesId DESC LIMIT 1)
        WHERE (SELECT usuarioId FROM interacciones ORDER BY interaccionesId
        DESC LIMIT 1)=i.usuarioId //
DELIMITER ;
```

Añadir datos a historial depreciaciones

```
DELIMITER //
CREATE OR REPLACE TRIGGER ratio_depreciacion
AFTER INSERT ON compras
FOR EACH ROW
BEGIN
        INSERT INTO historial_depreciaciones VALUES
        (historial_depreciaciones.depreciacionId, (SELECT fechaVent FROM compras
        NATURAL JOIN productos
        WHERE compras.comprald=productos.comprald ORDER BY comprald LIMIT 1),
        (SELECT ((SELECT precio FROM compras NATURAL JOIN productos
        ORDER BY comprald DESC LIMIT 1)/((SELECT DATEDIFF(NOW(),(SELECT
        fechaVent FROM compras NATURAL JOIN productos
        WHERE compras.comprald=productos.comprald ORDER BY comprald LIMIT
        1)))/90)) FROM compras NATURAL JOIN productos LIMIT 1),
        (SELECT (SELECT productId FROM productos NATURAL JOIN compras WHERE
        (SELECT comprald FROM compras ORDER BY comprald DESC LIMIT
        1)=productos.comprald)));
        END //
DELIMITER ;
```

Actualizar tabla de productos

```
DELIMITER //
CREATE OR REPLACE TRIGGER cambio_productos
AFTER INSERT ON compras
FOR EACH ROW
BEGIN
        INSERT INTO productos VALUES ((SELECT productoId+1 FROM productos
        ORDER BY productoId DESC LIMIT 1), (SELECT codigoEAN FROM productos
        NATURAL JOIN compras
        WHERE (SELECT juegoId FROM juegos NATURAL JOIN productos WHERE
        ((SELECT juegoId FROM productos
        NATURAL JOIN juegos ORDER BY compraId DESC LIMIT 1)=productos.juegoId))
ORDER BY compraId DESC LIMIT 1) ,
        (SELECT stock-1 FROM productos NATURAL JOIN compras WHERE
                (SELECT productoId FROM compras ORDER BY compraId DESC LIMIT
        1)=productos.productoId ORDER BY compraId DESC LIMIT 1),
                (SELECT compraId FROM productos NATURAL JOIN compras WHERE
        (SELECT productoId FROM compras ORDER BY compraId
                DESC LIMIT 1)=productos.productoId ORDER BY compraId DESC LIMIT 1),
        (SELECT juegoId FROM juegos NATURAL JOIN productos WHERE (SELECT
        juegoId
        FROM productos ORDER BY productoId DESC LIMIT 1)=juegos.juegoId LIMIT 1));

        END //
DELIMITER ;
```