



	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

Entregar un enlace a github al producto desarrollado en cada una de las fases del proyecto, y un archivo zip con el proyecto git.

El Readme de Github debe incluir una descripción de la GENERACIÓN DE CÓDIGO DE COMPONENTES Y PROCEDIMIENTOS (CSI-2)

CÓDIGO DE COMPONENTES
<pre> -- ----- -- Tabla Usuario -- ----- DROP TABLE IF EXISTS Usuario CASCADE; CREATE TABLE IF NOT EXISTS Usuario (Email VARCHAR(45) NOT NULL, Contraseña VARCHAR(45) NOT NULL, Nombre VARCHAR(45) NOT NULL UNIQUE, Imagen BYTEA NULL, PRIMARY KEY (Email)); -- ----- -- Tabla Basico -- ----- DROP TABLE IF EXISTS Basico CASCADE; CREATE TABLE IF NOT EXISTS Basico (Email VARCHAR(45) NOT NULL, Contraseña VARCHAR(45) NOT NULL, Nombre VARCHAR(45) NOT NULL UNIQUE, Imagen BYTEA NULL, PRIMARY KEY (Email)); -- ----- -- Tabla No_Basico -- ----- DROP TABLE IF EXISTS No_Basico CASCADE; CREATE TABLE IF NOT EXISTS No_Basico (Email VARCHAR(45) NOT NULL, Contraseña VARCHAR(45) NOT NULL, Nombre VARCHAR(45) NOT NULL UNIQUE, Imagen BYTEA NULL, Tipo VARCHAR(7) NULL CHECK (Tipo = 'Premium' OR Tipo = 'Deluxe'), PRIMARY KEY (Email)); </pre>

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

```

-- -----
-- Tabla Categoria
-- -----
DROP TABLE IF EXISTS Categoria CASCADE;

CREATE TABLE IF NOT EXISTS Categoria (
  Nombre VARCHAR(50) NOT NULL,
  Num_Titulos INT NULL,

  PRIMARY KEY (Nombre)
);

-- -----
-- Tabla Videojuego
-- -----
DROP TABLE IF EXISTS Videojuego CASCADE;

CREATE TABLE IF NOT EXISTS Videojuego (
  Distribuidora VARCHAR(45) NOT NULL,
  Nombre VARCHAR(45) NOT NULL,
  Año DATE NOT NULL CHECK (Año > 'Jan-01-1950' AND Año < CURRENT_DATE),
  PEGI INT NULL CHECK (PEGI IN (3, 7, 12, 16, 18)),


  PRIMARY KEY (Distribuidora, Nombre, Año)
);

-- -----
-- Tabla Externo
-- -----
DROP TABLE IF EXISTS Externo CASCADE;

CREATE TABLE IF NOT EXISTS Externo (
  Distribuidora VARCHAR(45) NOT NULL,
  Nombre VARCHAR(45) NOT NULL,
  Año DATE NOT NULL CHECK (Año > 'Jan-01-1950' AND Año < CURRENT_DATE),
  PEGI INT NULL CHECK (PEGI IN (3, 7, 12, 16, 18)),
  Desarrolladora VARCHAR(45) NULL,

  PRIMARY KEY (Distribuidora, Nombre, Año)
);

```

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

```

-----
-- Tabla De_Indev
-----
DROP TABLE IF EXISTS De_Indev CASCADE;

CREATE TABLE IF NOT EXISTS De_Indev (
  Distribuidora VARCHAR(45) NOT NULL,
  Nombre VARCHAR(45) NOT NULL,
  Año DATE NOT NULL CHECK (Año > 'Jan-01-1950' AND Año < CURRENT_DATE),
  PEGI INT NULL CHECK (PEGI IN (3, 7, 12, 16, 18)),
  Tipo VARCHAR(11) NULL CHECK (Tipo = 'Reciente' OR Tipo = 'No Reciente'),

  PRIMARY KEY (Distribuidora, Nombre, Año)
);


-----
-- Tabla Copia_Fisica
-----
DROP TABLE IF EXISTS Copia_Fisica CASCADE;

CREATE TABLE IF NOT EXISTS Copia_Fisica (
  Serial SERIAL UNIQUE NOT NULL,
  Distribuidora VARCHAR(45) NOT NULL,
  Nombre_Videojuego VARCHAR(45) NOT NULL,
  Año DATE NOT NULL CHECK (Año > 'Jan-01-1950' AND Año < CURRENT_DATE),

  PRIMARY KEY (Serial, Distribuidora, Nombre_Videojuego, Año),

  CONSTRAINT fk_Videojuego_CopiaFisica
  FOREIGN KEY (Distribuidora, Nombre_Videojuego, Año)
  REFERENCES Videojuego (Distribuidora, Nombre, Año)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION
);

```

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

```

-- -----
-- Tabla Filtra
-- -----
DROP TABLE IF EXISTS Filtra CASCADE;

CREATE TABLE IF NOT EXISTS Filtra (
  Email_Usuario VARCHAR(45) NOT NULL,
  Nombre_Categoria VARCHAR(50) NOT NULL,
  Fecha DATE NOT NULL,

  PRIMARY KEY (Email_Usuario, Nombre_Categoria, Fecha),

  CONSTRAINT fk_Usuario_Filtra
  FOREIGN KEY (Email_Usuario)
  REFERENCES Usuario (Email)
  ON DELETE CASCADE
  ON UPDATE CASCADE,

  CONSTRAINT fk_Categoria_Filtra
  FOREIGN KEY (Nombre_Categoria)
  REFERENCES Categoria (Nombre)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);

-- -----
-- Tabla Pertenece
-- -----
DROP TABLE IF EXISTS Pertenece CASCADE;


CREATE TABLE IF NOT EXISTS Pertenece (
  Distribuidora VARCHAR(45) NOT NULL,
  Nombre_Videojuego VARCHAR(45) NOT NULL,
  Año DATE NOT NULL CHECK (Año > 'Jan-01-1950' AND Año < CURRENT_DATE),
  Nombre_Categoria VARCHAR(50) NOT NULL,

  PRIMARY KEY (Distribuidora, Nombre_Videojuego, Año, Nombre_Categoria),

  CONSTRAINT fk_Videojuego_Pertenece
  FOREIGN KEY (Distribuidora, Nombre_Videojuego, Año)
  REFERENCES Videojuego (Distribuidora, Nombre, Año)
  ON DELETE CASCADE
  ON UPDATE CASCADE,

  CONSTRAINT fk_Categoria_Pertenece
  FOREIGN KEY (Nombre_Categoria)
  REFERENCES Categoria (Nombre)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);

```

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

```

-- -----
-- Tabla Recibe
-- -----
DROP TABLE IF EXISTS Recibe CASCADE;

CREATE TABLE IF NOT EXISTS Recibe (
    Serial_CpFisica INT NOT NULL,
    Email_NoBasico VARCHAR(45) NOT NULL,

    PRIMARY KEY (Serial_CpFisica, Email_NoBasico),

    CONSTRAINT fk_CpFisica_Recibe
    FOREIGN KEY (Serial_CpFisica)
    REFERENCES Copia_Fisica (Serial)
    ON DELETE CASCADE
    ON UPDATE CASCADE,

    CONSTRAINT fk_NoBasico_Recibe
    FOREIGN KEY (Email_NoBasico)
    REFERENCES No_Basico (Email)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

-- -----
-- Tabla Juegal
-- -----
DROP TABLE IF EXISTS Juegal CASCADE;


CREATE TABLE IF NOT EXISTS Juegal (
    Email_Basico VARCHAR(45) NOT NULL,
    Distribuidora VARCHAR(45) NOT NULL,
    Nombre_Externo VARCHAR(45) NOT NULL,
    Año DATE NOT NULL CHECK (Año > 'Jan-01-1950' AND Año < CURRENT_DATE),

    PRIMARY KEY (Email_Basico, Distribuidora, Nombre_Externo, Año),

    CONSTRAINT fk_Basico_Juegal
    FOREIGN KEY (Email_Basico)
    REFERENCES Basico (Email)
    ON DELETE CASCADE
    ON UPDATE CASCADE,

    CONSTRAINT fk_Externo_Juegal
    FOREIGN KEY (Distribuidora, Nombre_Externo, Año)
    REFERENCES Externo (Distribuidora, Nombre, Año)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

```

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

```

-- -----
-- Tabla Juega2
-- -----
DROP TABLE IF EXISTS Juega2 CASCADE;

CREATE TABLE IF NOT EXISTS Juega2 (
  Email_Basico VARCHAR(45) NOT NULL,
  Distribuidora VARCHAR(45) NOT NULL,
  Nombre_DeIndev VARCHAR(45) NOT NULL,
  Año DATE NOT NULL CHECK (Año > 'Jan-01-1950' AND Año < CURRENT_DATE),

  PRIMARY KEY (Email_Basico, Distribuidora, Nombre_DeIndev, Año),

  CONSTRAINT fk_Basico_Juega2
  FOREIGN KEY (Email_Basico)
  REFERENCES Basico (Email)
  ON DELETE CASCADE
  ON UPDATE CASCADE,

  CONSTRAINT fk_DeIndev_Juega1
  FOREIGN KEY (Distribuidora , Nombre_DeIndev , Año)
  REFERENCES De_Indev (Distribuidora , Nombre , Año)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);

-- -----
-- Tabla Juega3
-- -----
DROP TABLE IF EXISTS Juega3 CASCADE;


CREATE TABLE IF NOT EXISTS Juega3 (
  Email_NoBasico VARCHAR(45) NOT NULL,
  Distribuidora VARCHAR(45) NOT NULL,
  Nombre_Videojuego VARCHAR(45) NOT NULL,
  Año DATE NOT NULL CHECK (Año > 'Jan-01-1950' AND Año < CURRENT_DATE),

  PRIMARY KEY (Email_NoBasico, Distribuidora, Nombre_Videojuego, Año),


  CONSTRAINT fk_No_Basico_Juega3
  FOREIGN KEY (Email_NoBasico)
  REFERENCES No_Basico (Email)
  ON DELETE CASCADE
  ON UPDATE CASCADE,

  CONSTRAINT fk_Videojuego_Juega3
  FOREIGN KEY (Distribuidora , Nombre_Videojuego , Año)
  REFERENCES Videojuego (Distribuidora , Nombre , Año)

```

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

<pre> ON DELETE CASCADE ON UPDATE CASCADE); -- ----- -- Tabla Juega_Online ----- DROP TABLE IF EXISTS Juega_Online CASCADE; CREATE TABLE IF NOT EXISTS Juega_Online (Email_NoBasico1 VARCHAR(45) NOT NULL, Email_NoBasico2 VARCHAR(45) NOT NULL, Distribuidora VARCHAR(45) NOT NULL, Nombre_Videojuego VARCHAR(45) NOT NULL, Año DATE NOT NULL CHECK (Año > 'Jan-01-1950' AND Año < CURRENT_DATE), PRIMARY KEY (Email_NoBasico1, Distribuidora, Nombre_Videojuego, Año, Email_NoBasico2), CONSTRAINT fk_NoBasico1_JuegaOnline FOREIGN KEY (Email_NoBasico1) REFERENCES No_Basico (Email) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT fk_No_Basico2 FOREIGN KEY (Email_NoBasico2) REFERENCES No_Basico (Email) ON DELETE CASCADE ON UPDATE CASCADE, CONSTRAINT fk_Videojuego_JuegaOnline FOREIGN KEY (Distribuidora, Nombre_Videojuego , Año) REFERENCES Videojuego (Distribuidora , Nombre , Año) ON DELETE CASCADE ON UPDATE CASCADE); </pre>

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

CÓDIGO DE PROCEDIMIENTOS DE OPERACIÓN Y SEGURIDAD

Comprobar la inserción en Basico y No_Basico

En el modelo, en ambas relaciones IS_A las subtablas no pueden compartir entradas entre ellas. Esto se comprueba mediante los triggers **Check_Basico_Insert** y **Check_NoBasico_Insert**:

```

-- -----
-- Función y Trigger Basico_Insert
-- Las categorías de usuarios son excluyentes.
-- Es decir, un usuario BASICO no puede ser NO_BASICO.
-- -----
DROP FUNCTION IF EXISTS Basico_Insert() CASCADE;

CREATE FUNCTION Basico_Insert() RETURNS TRIGGER AS $$
BEGIN
    IF (NEW.Email IN (SELECT Email
                      FROM No_Basico))
    THEN
        RAISE 'Usuario % repetido.', NEW.Email
        USING HINT = 'Un usuario no puede ser Básico y No_Básico a la vez.';
    END IF;


    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Check_Basico_Insert
BEFORE INSERT ON Basico FOR EACH ROW EXECUTE PROCEDURE Basico_Insert();


-- -----
-- Función y Trigger NoBasico_Insert
-- Las categorías de usuarios son excluyentes.
-- Es decir, un usuario BASICO no puede ser NO_BASICO.
-- -----
DROP FUNCTION IF EXISTS NoBasico_Insert() CASCADE;

CREATE FUNCTION NoBasico_Insert() RETURNS TRIGGER AS $$
BEGIN
    IF (NEW.Email IN (SELECT Email
                      FROM Basico))
    THEN
        RAISE 'Usuario % repetido.', NEW.Email
        USING HINT = 'Un usuario no puede ser Básico y No_Básico a la vez.';
    END IF;

```


	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

<pre> RETURN NEW; END; \$\$ LANGUAGE plpgsql; CREATE TRIGGER Check_NoBasico_Insert BEFORE INSERT ON No_Basico FOR EACH ROW EXECUTE PROCEDURE NoBasico_Insert(); Ambos triggers hacen lo mismo: comprueban que los datos insertados no están en la tabla contraria antes de insertarlos y, si lo están lanzan un mensaje de error. Además, también se modelizó que todos los usuarios de Basico y No_Basico debían estar en Usuario. Esto se comprueba en la inserción de ambas subtablas por los triggers Basico_EstaEn_Usuario_Trigger y NoBasico_EstaEn_Usuario_Trigger. -- ----- -- Función y Trigger Basico_EstaEn_Usuario -- Todos los usuarios de las relaciones BASICO y -- NO_BASICO tienen que estar en USUARIO. -- ----- DROP FUNCTION IF EXISTS Basico_EstaEn_Usuario() CASCADE; CREATE FUNCTION Basico_EstaEn_Usuario() RETURNS TRIGGER AS \$\$ BEGIN IF (NEW.Email NOT IN (SELECT Email FROM Usuario WHERE Pago IS NULL)) THEN RAISE 'Usuario no admitido.' USING HINT = 'Todos los usuarios tienen que estar en la tabla Usuarios con su Pago indicado antes que en cualquier otra.'; END IF; RETURN NEW; END; \$\$ LANGUAGE plpgsql; CREATE TRIGGER Basico_EstaEn_Usuario_Trigger BEFORE INSERT ON Basico FOR EACH ROW EXECUTE PROCEDURE Premium_EstaEn_Usuario(); -- ----- -- Función y Trigger Premium_EstaEn_Usuario -- Todos los usuarios de las relaciones BASICO y -- NO_BASICO tienen que estar en USUARIO. -- ----- DROP FUNCTION IF EXISTS NoBasico_EstaEn_Usuario() CASCADE; CREATE FUNCTION NoBasico_EstaEn_Usuario() RETURNS TRIGGER AS \$\$ BEGIN IF (NEW.Email NOT IN (SELECT Email FROM Usuario WHERE Pago >= 1)) THEN RAISE 'Usuario no admitido.' </pre>
--

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

```

        USING HINT = 'Todos los usuarios tienen que estar en la tabla Usuarios con su
Pago indicado antes que en cualquier otra.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

```

CREATE TRIGGER NoBasico_EstaEn_Usuario_Trigger
BEFORE INSERT ON No_Basico FOR EACH ROW EXECUTE PROCEDURE NoBasico_EstaEn_Usuario();

```

Sendos triggers realizan las mismas acciones: Comprueban si el Email insertado está en la lista de los de la tabla Usuario (comprobando en cada caso el pago que realizan para situarlos en su correspondiente tabla), y si no lo está lanza un mensaje de error.

Inserción de Usuarios en Basico y No_Basico

Se requiere que, al insertar un usuario en Usuario se compruebe si es Básico, Premium o Deluxe según el pago de su suscripción. Esto lo realiza el trigger Inserta_Usuario_Trigger.

```

-- -----
-- Función y Trigger Inserta_Usuario
-- El campo Pago de la relación USUARIO define el tipo
-- de suscripción de cada usuario: Un valor nulo indica
-- que el usuario es Básico, de 1 a 10 de No Básico con
-- Tipo 'Premium', y de 11 en adelante No Básico de
-- Tipo 'Deluxe'.
-- -----

DROP FUNCTION IF EXISTS Inserta_Usuario() CASCADE;


CREATE FUNCTION Inserta_Usuario() RETURNS TRIGGER AS $$
BEGIN
    IF (NEW.Pago IS NULL) THEN
        INSERT INTO Basico (Email, Contraseña, Nombre, Imagen)
        VALUES (NEW.Email, NEW.Contraseña, NEW.Nombre, NEW.Imagen);
    END IF;

    IF (NEW.Pago >= 1 AND NEW.Pago <= 10) THEN
        INSERT INTO No_Basico (Email, Contraseña, Nombre, Imagen, Tipo)
        VALUES (NEW.Email, NEW.Contraseña, NEW.Nombre, NEW.Imagen, 'Premium');
    END IF;

    IF (NEW.Pago >= 11) THEN
        INSERT INTO No_Basico (Email, Contraseña, Nombre, Imagen, Tipo)
        VALUES (NEW.Email, NEW.Contraseña, NEW.Nombre, NEW.Imagen, 'Deluxe');
    END IF;

    RETURN NEW;
END;

```

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

```
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER Inserta_Usuario_Trigger
AFTER INSERT ON Usuario FOR EACH ROW EXECUTE PROCEDURE Inserta_Usuario();
```

Comprobar la inserción en Copia_Fisica

Como se comentó anteriormente, solamente los usuarios No Básicos cuyo tipo sea Deluxe pueden recibir copias físicas de videojuegos. Esto se comprueba mediante el trigger Check_CpFisica_Insert:

```
-- -----
-- Función y Trigger CpFisica_Insert
-- Sólo los usuarios de la tabla NO_BASICO con Tipo
-- 'Deluxe' pueden estar en la tabla Copia_Fisica.
-- -----


DROP FUNCTION IF EXISTS CpFisica_Insert() CASCADE;

CREATE FUNCTION CpFisica_Insert() RETURNS TRIGGER AS $$
BEGIN
    IF ((SELECT Tipo
        FROM No_Basico
        WHERE Email = NEW.Email_NoBasico AND
              Tipo = 'Deluxe') IS NULL) THEN
        RAISE 'Usuario no admitido.'
        USING HINT = 'Sólo los usuarios Deluxe pueden recibir copias físicas.';
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Check_CpFisica_Insert
BEFORE INSERT ON Copia_Fisica EXECUTE PROCEDURE CpFisica_Insert();
```

Al insertar un registro en la tabla Copia_Fisica, selecciona el usuario con el email insertado y tipo Deluxe. Si la selección es nula, el tipo es incorrecto, por lo que lanza un mensaje de error.

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

Comprobar la inserción en Externo y De_Index

Las tablas Videojuego, Externo y De_Index tienen la misma relación que la de los usuarios, por lo que sus triggers de inserción hacen casi lo mismo: comprobar que no hay datos repetidos entre las tablas hijo.

```
-- -----
-- Función y Trigger Externo_Insert
-- Las categorías de videojuegos son excluyentes.
-- Es decir, un videojuego EXTERNO no puede ser DE_INDEV.
-- -----
DROP FUNCTION IF EXISTS Externo_Insert() CASCADE;


CREATE FUNCTION Externo_Insert() RETURNS TRIGGER AS $$
BEGIN
    IF ((SELECT Nombre
        FROM De_Index
        WHERE (Distribuidora = NEW.Distribuidora AND
              Nombre = NEW.Nombre AND
              Año = NEW.Año)) IS NOT NULL) THEN
        RAISE 'Videojuego repetido.'
        USING HINT = 'Un videojuego no puede ser Externo y De_Index a la vez.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Check_Externo_Insert
BEFORE INSERT ON Externo FOR EACH ROW EXECUTE PROCEDURE Externo_Insert();

-- -----
-- Función y Trigger DeIndex_Insert
-- Las categorías de videojuegos son excluyentes.
-- Es decir, un videojuego EXTERNO no puede ser DE_INDEV.
-- -----
DROP FUNCTION IF EXISTS DeIndex_Insert() CASCADE;

CREATE FUNCTION DeIndex_Insert() RETURNS TRIGGER AS $$
BEGIN
    IF ((SELECT Nombre
        FROM Externo
        WHERE (Distribuidora = NEW.Distribuidora AND
```

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

```

        Nombre = NEW.Nombre AND
        Año = NEW.Año)) IS NOT NULL) THEN
    RAISE 'Videojuego repetido.'
    USING HINT = 'Un videojuego no puede ser Externo y De_Indev a la vez.';
END IF;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Check_DeIndev_Insert
BEFORE INSERT ON De_Indev FOR EACH ROW EXECUTE PROCEDURE DeIndev_Insert();
Y , al igual que con los usuarios, también tiene triggers para evitar la inserción de elementos en las tablas hijas
que no estén en la tabla padre.


-- -----
-- Función y Trigger Externo_EstaEn_Videojuego
-- Todos los Videojuegos de las relaciones EXTERNO y
-- DE_INDEV tienen que estar en VIDEOJUEGO.
-- -----
DROP FUNCTION IF EXISTS Externo_EstaEn_Videojuego() CASCADE;
CREATE FUNCTION Externo_EstaEn_Videojuego() RETURNS TRIGGER AS $$
BEGIN
    IF ((SELECT Nombre
        FROM Videojuego
        WHERE Distribuidora = NEW.Distribuidora AND
            Año = NEW.Año AND
            Nombre = NEW.Nombre AND
            Desarrolladora = NEW.Desarrolladora) IS NULL) THEN
        RAISE 'Videojuego no admitido.'
        USING HINT = 'Todos los videojuegos tienen que estar en la tabla Videojuegos con
su Desarrolladora indicada antes que en cualquier otra.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Externo_EstaEn_Videojuego_Trigger
BEFORE INSERT ON Externo FOR EACH ROW EXECUTE PROCEDURE Externo_EstaEn_Videojuego();

-- -----
-- Función y Trigger DeIndev_EstaEn_Videojuego
-- Todos los Videojuegos de las relaciones EXTERNO y
-- DE_INDEV tienen que estar en VIDEOJUEGO.
-- -----
DROP FUNCTION IF EXISTS DeIndev_EstaEn_Videojuego() CASCADE;
CREATE FUNCTION DeIndev_EstaEn_Videojuego() RETURNS TRIGGER AS $$
BEGIN
    IF ((SELECT Nombre
        FROM Videojuego
        WHERE Distribuidora = NEW.Distribuidora AND
            Año = NEW.Año AND

```

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

```

        Nombre = NEW.Nombre AND
        Desarrolladora IS NULL) IS NULL) THEN
        RAISE 'Videojuego no admitido.'
        USING HINT = 'Todos los videojuegos tienen que estar en la tabla Videojuegos con
su Desarrolladora indicada antes que en cualquier otra.';
        END IF;

        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

```

```

CREATE TRIGGER DeIndev_EstaEn_Videojuego_Trigger
BEFORE INSERT ON De_Indev FOR EACH ROW EXECUTE PROCEDURE DeIndev_EstaEn_Videojuego();

```

Inserción de Videojuegos en Externo y De_Indev

Se requiere que, al insertar un videojuego en Videojuego se compruebe si es Externo, De Indev Reciente o No Reciente según si se indica la desarrolladora y su fecha de lanzamiento. Esto lo realiza el trigger Inserta_Videojuego_Trigger.

```

-- -----
-- Función y Trigger Inserta_Videojuego
-- El campo Desarrolladora de Videojuego indica si es
-- un juego Externo o De Indev, y su Año de lanzamiento
-- si es reciente o no.
-- -----

DROP FUNCTION IF EXISTS Inserta_Videojuego() CASCADE;

CREATE FUNCTION Inserta_Videojuego() RETURNS TRIGGER AS $$
BEGIN
    IF (NEW.Desarrolladora IS NULL) THEN
        IF (CURRENT_DATE - NEW.Año <= 30) THEN
            INSERT INTO De_Indev (Distribuidora, Nombre, Año, Tipo)
            VALUES (NEW.Distribuidora, NEW.Nombre, NEW.Año, 'Reciente');

        ELSE
            INSERT INTO De_Indev (Distribuidora, Nombre, Año, Tipo)
            VALUES (NEW.Distribuidora, NEW.Nombre, NEW.Año, 'No Reciente');
        END IF;


    ELSE
        INSERT INTO Externo (Desarrolladora, Distribuidora, Nombre, Año)
        VALUES (NEW.Desarrolladora, NEW.Distribuidora, NEW.Nombre, NEW.Año);
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Inserta_Videojuego_Trigger
AFTER INSERT ON Videojuego FOR EACH ROW EXECUTE PROCEDURE Inserta_Videojuego();

```

Si no se indica la desarrolladora (valor nulo), inserta al videojuego en De_Indev. Si han pasado más de 30 días

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

desde la publicación del título, se inserta con el campo Tipo a 'No Reciente', y si no a 'Reciente'. Y si se indica el nombre de la desarrolladora, se inserta en Externo.

Comprobar la inserción en Juega1 y Juega2


Para mantener la condición de exclusión se estableció en el modelo que los usuarios Básicos sólo pueden jugar a videojuegos Externos o De_Indev de tipo Reciente, teniendo que elegir entre uno de estas categorías; y siendo el primer videojuego al que jugasen el que la definiría.

Esto se lograba estableciendo dos tablas (Juega1 y Juega2). La primera contiene sólo videojuegos Externos, y la segunda sólo De Indev. Para mantener la condición de exclusividad se debe comprobar que, a la hora de insertar datos en cualquiera de las dos tablas, no haya usuarios duplicados. Mediante los triggers Check_Juega1_Insert y Check_Juega2_Insert se consigue.

```
-- -----
-- Función y Trigger Juegal_Insert
-- No puede haber usuarios repetidos entre las
-- relaciones JUEGA1 y JUEGA2.
-- -----
DROP FUNCTION IF EXISTS Juegal_Insert() CASCADE;
CREATE FUNCTION Juegal_Insert() RETURNS TRIGGER AS $$
BEGIN
    IF (NEW.Email_Basico IN (SELECT Email_Basico
                           FROM Juega2))
    THEN
        RAISE 'Usuario no admitido.'
        USING HINT = 'Un usuario sólo puede jugar a videojuegos EXTERNOS o DE_INDEV de
tipo "Reciente", eligiendo entre un tipo u otro.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Check_Juegal_Insert
BEFORE INSERT ON Juega1 FOR EACH ROW EXECUTE PROCEDURE Juegal_Insert()
```

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

```

-- -----
-- Función y Trigger Juega2_Insert
-- No puede haber usuarios repetidos entre las
-- relaciones JUEGA1 y JUEGA2.
-- En la tabla JUEGA2 sólo puede haber videojuegos de
-- la relación DE_INDEV cuyo Tipo sea 'No Reciente'.
-- -----
DROP FUNCTION IF EXISTS Juega2_Insert() CASCADE;


CREATE FUNCTION Juega2_Insert() RETURNS TRIGGER AS $$
BEGIN
    IF (NEW.Email_Basico IN (SELECT Email_Basico
                             FROM Juega1))
    THEN
        RAISE 'Usuario no admitido.'
        USING HINT = 'Un usuario sólo puede jugar a videojuegos EXTERNOS o DE_INDEV de
tipo "Reciente", eligiendo entre un tipo u otro.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER Check_Juega2_Insert
BEFORE INSERT ON Juega2 FOR EACH ROW EXECUTE PROCEDURE Juega2_Insert();

```

Ambos disparadores hacen lo mismo: Comprueban que el usuario introducido no se encuentra en la tabla contraria y, si está, lanza un mensaje de error.

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

Actualizar la cuenta de títulos por categoría

Al asignarle nuevas categorías a videojuegos mediante la tabla Pertenece se debe de actualizar la cuenta de títulos por categoría de la tabla Categoría. Esto se consigue mediante el trigger Actualiza_Categoría_Trigger.


```
-- -----
-- Función y Trigger Actualiza_Categoría
-- Cada vez que se asigna un videojuego a una
-- categoría, se actualiza la cuenta de videojuegos
-- por categoría.
-- -----
DROP FUNCTION IF EXISTS Cuenta_Titulos() CASCADE;

CREATE FUNCTION Cuenta_Titulos() RETURNS INTEGER AS $count$
BEGIN
    RETURN (SELECT COUNT(*)::int
            FROM Pertenece
            WHERE Nombre_Categoría = NEW.Nombre_Categoría);
END;
$count$ LANGUAGE plpgsql;

DROP FUNCTION IF EXISTS Actualiza_Categoría() CASCADE;

CREATE FUNCTION Actualiza_Categoría() RETURNS TRIGGER AS $$
BEGIN
    UPDATE Categoría
    SET Num_Titulos = (SELECT COUNT(*)::int
                      FROM Pertenece
                      WHERE Nombre_Categoría = NEW.Nombre_Categoría)
    WHERE (Nombre = NEW.Nombre_Categoría);

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

	CONSTRUCCIÓN DEL SISTEMA DE INFORMACIÓN (CSI)	BASES DE DATOS
 Escuela Superior de Ingeniería y Tecnología Universidad de La Laguna	PROYECTO: Servicio de suscripción de Indev	Generación de código
	Autores: Jaime Simeón Palomar Blumenthal Alberto Cruz Luis Cristo García González Antonella Sofía García Álvarez	ALU0101228587 ALU0101217734 ALU0101204512 ALU0101227610
Versión: 0.1.0	Ref: SCRIPT	Tiempo invertido: 16 horas
		Fecha : 01/02/22

```
CREATE TRIGGER Actualiza_Categoria_Trigger
AFTER INSERT ON Pertenece FOR EACH ROW EXECUTE PROCEDURE Actualiza_Categoria();
```

Al introducir un nuevo registro en la tabla Pertenece se cuenta el número de filas con el nombre de la categoría que se ha utilizado para la inserción, y se actualiza la tabla categoría estableciendo en el registro de la categoría de la inserción el atributo Num_Títulos al valor de la cuenta hecha previamente.