Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Proyecto: myLibrary

Administración y Diseño de Bases de datos

adri puton

Asignatura: Administración y Diseño de Bases de Datos
Curso: Primer cuatrimestre, 4º año, grado en Ingeniería informática
Integrantes del grupo: Adrián González Galván MARCOS PADILLA HERRERA
Fecha de entrega: -

C/ Padre Herrera s/n
38207 La Laguna
Santa Cruz de Tenerife. España

T: 900 43 25 26

ull.es

Índice

Introducción

El siguiente documento recoge información acerca del **proyecto final** de la asignatura de 4º curso del grado de Ingeniería Informática de la Universidad de la Laguna: **“Administración y Diseño de Bases de Datos”**.

Este proyecto plantea la realización **del diseño y la implementación de una base de datos** partiendo desde cero. Esto, por consiguiente, conlleva el planteamiento de una estructura de datos, la creación de modelos “relacional” y “entidad-relación”, la utilización de algún gestor de bases de datos y el manejo de algún framework para desarrollo web con el que realizar peticiones a una API. En nuestro caso, las **herramientas base** que emplearemos serán Draw.io, PostgreSQL y Flask.

El correspondiente enunciado de la actividad lo podremos encontrar en el siguiente enlace.

Base de datos implementada

Título y descripción

La base de datos que se ha construido para el desarrollo de este proyecto tiene como título **“myLibrary”**.

La base de datos realizada simula ser un caso particular de una **base de datos** utilizada en una **biblioteca** dentro de la comunidad de Canarias. Esta es una base de datos de tipo SQL compuesta por un conjunto de 23 tablas, de entre las que destacan las siguientes:

- Artículos: Los cuales pueden ser desde libros hasta revistas, periódicos, documentales, series o películas. Para ello se han categorizado los tipos de artículos en 3: libros, materiales periódicos y materiales audiovisuales.
- Usuarios: Los cuales adquieren el material de la biblioteca. Para estos se ha tenido en cuenta que puedan ser mayores o menores de edad.
- Trabajadores: Quienes son los encargados de trabajar con esta base de datos para el registro tanto de los usuarios, como de los artículos y como de las prestaciones que vayan a realizarse.
- Prestaciones: Las cuales son el principal eje de la base de datos. Estas tienen un identificador único, una fecha de inicio y una fecha de finalización, entre otras cosas. El registro de estas es

útil para, entre otros, establecer un límite en el número de prestaciones.

Es importante remarcar que además de estas existen otras tablas o entidades también empleadas en la base de datos. Estas entidades no son tan relevantes puesto que simplemente complementan la información de las denominadas principales. Estas son: isla, municipio, dirección, horario, día y periodo. Además de otras derivadas de atributos multivaluados como podrían ser los de “teléfono” o “email”. No obstante, visualizaremos la estructura de todas estas a través del modelo entidad relación que comentaremos más adelante.

La idea, tal y como se ha comentado describiendo a la entidad ‘Trabajadores’, es que esta base de datos sea **empleada** principalmente **por los trabajadores de la biblioteca** donde se utilice. El planteamiento expuesto sugiere que serán los mismos trabajadores los que emplearán la base de datos para registrar los artículos presentes en la biblioteca, para incorporar los datos de los nuevos usuarios que quieran recibir prestaciones y para crear la información de las prestaciones correspondientes. Si bien los mismo trabajadores no deberían ser capaces de administrar los propios registro de la tabla ‘trabajadores’, esta se ha creado con el principal objetivo de plasmar en cada una de las prestaciones realizadas quien fue el trabajador que la realizó.

Concluyendo, se podría redondear que el proyecto se basa en la creación de una base de datos empleada para almacenar información relativa a una biblioteca y manejada por los trabajadores de la misma. Así pues, la idea es que sea una **base de datos organizacional** que empleen los trabajadores desde sus mismas oficinas y a la que puedan mandar peticiones para registrar y manejar información relativa a los artículos, los usuarios y las prestaciones.

Objetivos que pretende abarcar la base de datos

La base de datos diseñada se ha implementado conforme a una serie de objetivos que se deseaban satisfacer. Entre ellos destacan:

- ☐ **Digitalizar todos los datos** existentes del conjunto de artículos presentes en una biblioteca, los cuales pueden alcanzar un tamaño considerable.
- ☐ Disponer de un **catálogo de artículos** fácil de emplear y visualizar.

- ☐ Determinar la **disponibilidad** (número de unidades) **de cada uno de los artículos** de la biblioteca.
- ☐ Llevar un **registro** de los datos de todos los **usuarios** adscritos a la biblioteca, así como de los **trabajadores** de la misma.
- ☐ Facilitar a los trabajadores el **manejo** de la información sobre las **prestaciones** realizadas.

El principal propósito que se desea satisfacer es el de **agilizar** de alguna manera **el proceso de trabajo** de los empleados de la biblioteca. Empleando una base de datos de este estilo será sencillo para los trabajadores realizar diversas tareas que impliquen consultar la disponibilidad de determinados artículos o comprobar el estado de algunas prestaciones en particular. Al tener toda esta información de una manera centralizada, como lo es a través de un sistema informático, resulta más sencillo elaborar tareas de gestión.

Sabemos que la cantidad de volúmenes de datos con respecto a los artículos presentes en una biblioteca puede llegar a ser demasiado grande. Así pues, tener un registro de las prestaciones realizadas puede llegar a ser todo un reto puesto que se deben plasmar los usuarios que las realizaron, las fechas en las que finalizan, las fechas en las que finalmente son devuelta; además de consultar la vigencia de las mismas. Con todo ello, el planteamiento de una base de datos como la que proponemos puede ser clave en la gestión de una organización así.

Especificación de requisitos

Como se ha comentado, la **principal función** que tiene la base de datos implementada es la de ser **utilizada por los trabajadores** de una biblioteca como medio para **registrar, actualizar y eliminar usuarios y artículos** de la misma y para **manejar el conjunto de prestaciones** que pudieran solicitarse.

A la hora de construirla, la hemos diseñado atendiendo a una serie de requisitos. Tal y como se especificaba en el guión era necesario que la base de datos no prescindiera de entidades débiles, de casos IS_A o de relaciones triples, entre otras cosas. El **contexto** de la misma se puede apreciar a continuación:

1. **Entidades principales**
    a. **Usuario**

i. Esta representa al **usuario que va a disfrutar de los servicios de prestación** de artículos que presenta la biblioteca.

ii. Posee un identificador inequívoco que lo identifica, así como otra serie de datos como su nombre y apellidos, su dirección, su fecha de nacimiento, su edad, sus posibles teléfonos, etc.

iii. Hemos diferenciado que los usuarios puedan ser tanto **adultos** como **menores**.

b. Trabajador

i. Esta entidad representa al respectivo **trabajador de la biblioteca**.

ii. A diferencia del usuario, esta entidad se identifica a través de un DNI. No obstante, también contiene campos idénticos como el nombre completo, los telefonos o el sexo.

c. Artículo

i. Esta entidad es sin duda la más relevante. Esta representa al **conjunto de artículos** disponibles dentro de la biblioteca.

ii. Los artículos poseen un identificador único y una serie de atributos comunes independientemente de cual sea su **tipo (libro, material periodico o material audiovisual)** como el título, el posible subtítulo, la fecha de publicación o la portada.

d. Prestación

i. Esta entidad es otra de las más importantes junto con 'artículo', puesto que realiza la función principal de la organización: manejar prestaciones. Esta representa, por tanto, al **conjunto de prestaciones** de la biblioteca.

ii. Posee de igual manera un identificador único y está enlazada con las otras 3 entidades descritas anteriormente. Así pues, en cada prestación figuran el usuario que la solicitó, el trabajador que la concedió y el artículo que se quiere solicitar. Se podría decir que es la entidad que ancla todo el sistema.

2. Otras entidades y entidades débiles

a. Autor

i. Esta entidad representa al **conjunto de autores** con sus respectivos datos **que pueden tener los distintos artículos**.

ii. Los autores se encuentran asociados a artículos que bien sean libros o algún material audiovisual. Consideramos, para nuestro supuesto, que un periódico no tuviese autor.

iii. Esta entidad **no es débil**. Se considera que el registro de autores debe guardarse independientemente de si existen artículos asociados a ellos.

b. Tarjeta socio

i. Esta es una entidad simple con identificador y otro campo poco significativo (el color) y sirve para **identificar a un usuario menor** de edad.

ii. La principal motivación a la hora de crear esta entidad fue la de identificar a usuarios menores. Dado a que estos pueden no tener DNI que los identifique, sería apropiado que en un caso real tengan una tarjeta con un número con el que asociar sus datos.

iii. Es una **entidad débil** puesto que sin usuario menor asociado no tiene sentido almacenar la información de una tarjeta.

c. Dirección

i. A la hora de crear **usuarios** o **trabajadores** asumimos que se deben guardar las **direcciones físicas** de los mismos si es que tienen. En lugar de guardar estos datos como atributos se creó una entidad a parte ‘dirección’ con tal de organizar mejor los datos. Las direcciones tienen por tanto un identificador único.

ii. Tanto un trabajador como un usuario pueden tener múltiples direcciones.

iii. Esta es una **entidad débil**. Una dirección no existe si no tiene un trabajador o usuario asociado.

d. Isla

i. A la hora de crear una dirección vimos necesario **identificar la isla del archipiélago** a la que se encuentra asociada. Para ello es esta entidad isla, que identifica inequívocamente a una isla y proporciona otros datos de la misma como la georreferenciación.

ii. Esta **no es una entidad débil**.

e. Provincia

i. La entidad provincia representa a las posibles provincias que puede tener una isla y por consiguiente una dirección. Esta tiene un identificador inequívoco, aunque fundamentalmente son 2 las provincias a registrar: Santa Cruz de Tenerife y Las Palmas de Gran Canaria.

ii. Esta **no es una entidad débil**.

f. Horario

i. Esta entidad sirve a modo de establecer un **horario** para los distintos **trabajadores** de la biblioteca.

ii. Es una **entidad débil**: sin trabajador no existe horario.

g. Dia

i. Esta entidad deriva de horario y simplemente sirve para señalar cada uno de los posibles **días que puede tener un horario**.

ii. Es una **entidad débil** puesto que sin horario no pueden existir días de un horario.

h. Periodo

i. Con esta entidad se finaliza la definición de la tabla 'horario'. Esta representa los **distintos periodos** en los que un trabajador puede **trabajar** en un mismo día. Así, un trabajador puede tener el periodo '9:00-14:00' y al mismo tiempo el periodo '16:00-20:00' un día 'Lunes'.

ii. Es una entidad **débil** puesto que no existe sin un horario o día que lo defina.

i. Entidades de atributos multivaluados

i. Cabe también destacar aquellas entidades que se han generado a partir de atributos multivaluados de otras tablas. Estos son:

1. **Género:** Que representa los distintos géneros que puede tener un artículo.
2. **Teléfono:** Que representa los varios teléfonos asociados a un usuario o trabajador.
3. **Email:** Que representa los varios correos electrónicos asociados a un usuario o trabajador.

3. Relaciones

a. Algunas relaciones (1:1)

i. UsuarioMenor-TarjetaSocio: Un usuario menor posee una tarjeta socio con un identificador inequívoco. Y al mismo tiempo una tarjeta socio solo pertenece a un usuario menor. Como dijimos, sirve para identificar a un usuario menor en el caso de que no posea dni.

b. Algunas relaciones (1:n)

i. Usuario/Trabajador-Direccion: Un usuario o trabajador tiene múltiples direcciones, pero una dirección solo pertenece a un único usuario o trabajador.

ii. Libro-Autor: Un libro tiene un único autor, pero un autor puede estar asociado a varios libros.

iii. Artículo-Prestación: Un artículo puede tener asociadas ‘n’ prestaciones, pero una prestación sólo tiene asociado un único artículo.

c. Algunas relaciones (n:m)

i. MaterialAudiovisual-Autor: A diferencia de la entidad ‘libro’, un material audiovisual puede tener asociados múltiples

autores. Asimismo, un autor puede tener asociados múltiples material audiovisuales.

d. Algunas relaciones triples

Nuestro modelo no precisa relaciones triples. En un primer momento se pensaba crear una relación triple entre las entidades 'artículo', 'usuario' y 'trabajador' en la que se conformara lo que conocemos como **prestación**. Pero esta idea fue descartada puesto que, como si de una factura se tratase, sería mejor que las prestaciones dispusieran de un identificador único que las diferenciase. De este modo, sería mucho más sencillo almacenar la información de cada una de ellas a modo de 'registro'.

Por lo tanto, lo más parecido a una relación triple que pueda tener nuestro modelo sería la misma tabla 'prestación' que está conectada mediante otras relaciones a las entidades descritas en el párrafo anterior.

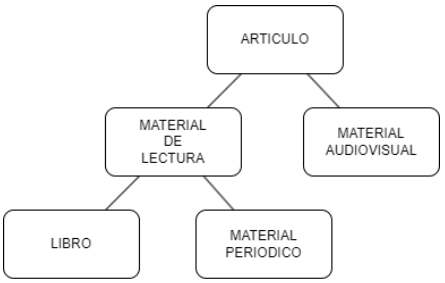
4. Tipos IS_A y casos de inclusión, inclusividad, exclusión y/o exclusividad

Para este caso, y de igual manera que con el caso anterior, nuestro modelo tampoco precisa de casos de inclusión, inclusividad, exclusión o exclusividad en cuanto a relaciones simples se refiere.

Sin embargo, cabe destacar que si tenemos **relaciones IS_A de tipo exclusiva y total**. Dado a que no vimos ningún caso en el cual fuera necesario el uso de relaciones con los casos mencionados, pensamos que la implementación de estos casos junto con las relaciones IS_A sería más que suficiente.

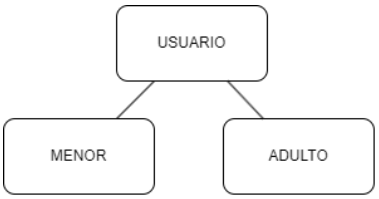
a. Artículo

i. Como ya hemos dicho , los artículos tienen un tipo. Esto es que los artículos pueden ser **libros**, **materiales periódicos** o **materiales audiovisuales**. La jerarquía formada tiene **3 niveles**: un primero donde se ubica el artículo, un segundo con las entidades 'material de lectura' y 'material audiovisual'; y un tercero desde el que derivan 'libro' y 'material periodico' de 'material de lectura.



b. Usuario

i. De esta entidad derivan otras 2 entidades: **usuario-adulto** y **usuario-menor**. La principal motivación de realizar esta división fue para aplicar una serie de ventajas a unos con respecto a otros atendiendo a una serie de factores



5. Supuestos semánticos

a. Usuario-prestaciones

i. Este es sin duda el supuesto semántico más importante del modelo. En este **restringimos el número de prestaciones que pueda realizar un usuario**. Entonces:

1. Si el **usuario es menor** puede realizar un total de 7 prestaciones máximas sin necesidad de devolver ningún artículo de los prestados.

2. Si es **adulto y es estudiante** sería el mismo caso.

3. Si es **adulto y no es estudiante** el número pasaría a 5.

ii. **La limitación infiere en el número de prestaciones no devueltas.** Es decir, un usuario puede realizar un número ilimitado de prestaciones. Pero tiene que devolver cada artículo. Si no devuelve el número se limita. Por ello, se ha registrado la fecha de devolución en cada prestación y una

variable que determina si la prestación se encuentra vigente o no.

b. Trabajador-periodos

Para el caso de los periodos de trabajo de un empleado distinguimos 2 apartados importantes:

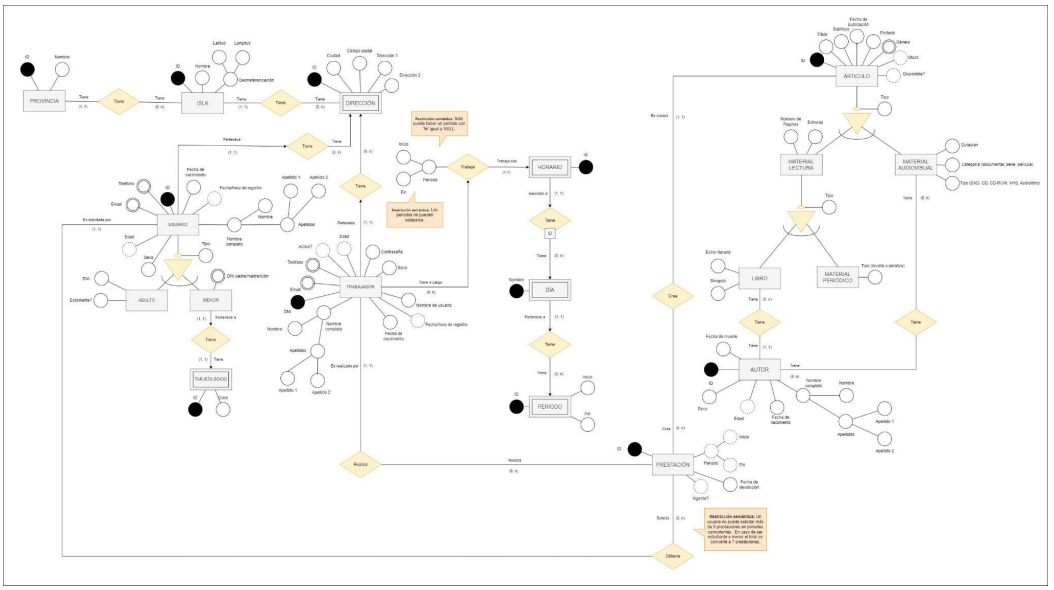
i. **Los periodos no pueden solaparse:** Un trabajador puede tener varios periodos de trabajo, sin embargo es de lógica que estos no se solapen ya que sería algo incoherente

ii. **No puede haber más de un periodo no finalizado:** Consideramos que un empleado está trabajando si alguno de sus periodos no se ha finalizado. Por lo tanto, tampoco sería coherente tener varios períodos no finalizados puesto que no puede tener más de un trabajo.

Modelo entidad-relación

Como se precisaba en el enunciado de la actividad del proyecto de la asignatura, se ha realizado un modelo entidad-relación con el objetivo de representar de una manera esquemática el diseño de la base de datos. Tanto las imágenes como el fichero en extensión 'drawio' se pueden encontrar en la carpeta del siguiente repositorio.

La imagen del modelo entidad relación creada es la siguiente:



En este modelo se pueden observar más a detalle la especificación de requisitos que comentamos anteriormente. Así pues, entre otras cosas:

- Se aprecian las relaciones descritas
- Se observa con mayor claridad la jerarquía en cuanto a las entidades artículo y usuario y sus casos de exclusividad.
- Se aprecian todos los atributos correspondientes con cada entidad.
- Se señalan las restricciones semánticas cerca de las entidades sobre las que intervienen.

Nótese que las restricciones semánticas expuestas son necesarias de señalar puesto que requieren de una serie de condicionantes que de otra manera serían imposibles de incluir en el modelo E-R.

Modelo relacional

El modelo relacional creado se ha realizado en un archivo de texto en formato '.md' dado a la simplicidad que aporta el mismo y el potencial que tiene para mostrar de manera clara cada uno de los datos de las tablas. Este se puede encontrar en el siguiente enlace.

Algunos de los ejemplos de tablas que podemos encontrar en este modelo son los siguientes:

1. **Artículo**

Articulo (ID, Tipo, Título, Subtítulo, Fecha_Publicación, Portada, Stock, Disponible)

- ID: PRIMARY KEY
- Tipo: CHECK IN [libro, materialPeriodico, materialAudiovisual]
- Titulo: NOT NULL
- Fecha_Publicacion <= FECHA_ACTUAL
- Stock: NOT NULL
- Stock > 0
- Stock TRIGGER actualizar cuando se cree o finalice una Prestacion
- Disponible TRIGGER TRUE : Stock > 0, FALSE : Stock = 0

Lo más significativo de esta tabla es quizás la **definición de un disparador** con el que se pretende **actualizar el stock** (aumentándolo o disminuyéndolo 1 unidad) de cada artículo al **crearse o finalizarse una prestación**.

2. **Género**

Genero (ID_Articulo, GeneroArticulo)

- ID_Articulo: FOREIGN KEY de Articulo(ID)
- GeneroArticulo: NOT NULL
- GeneroArticulo CHECK IN [Novela, Cuento, Poesía, Ensayo, Bi
  el-Tiempo, Misterio, Suspense, Detective, Thriller, Romance, A
  Terror, Infantil, Anime]

Esta es una de las **tablas realizadas a partir de atributos multi-valuados**. Como podemos ver se destaca que se ha remarcado la utilización de de atributos ajenos. Asimismo, se ha tipado el conjunto de géneros que puede tener un artículo.

3. Libro

Libro (ID_Articulo, Editorial, Número_Pag, Estilo, Sinopsis, ID_Autor)

- ID_Articulo: PRIMARY KEY AND FOREIGN KEY de Articulo(ID)
- ID_Autor: FOREIGN KEY de Autor(ID)
- Editorial: NOT NULL
- Numero_Pag: NOT NULL
- Numero_Pag > 0
- Estilo: NOT NULL
- Estilo CHECK IN [Narrativo, Poético, Dramático, Épico, Lírico, Didáctico, Satírico]

Para la tabla 'libro' resulta interesante remarcar que el **atributo ID es tanto primario como ajeno**.

4. Trabajador

Trabajador (DNI, Nombre, Apellido_1, Apellido_2, Fecha_Nacimiento, Nombre_Usuario, Sexo, Contraseña, Edad, Activo, Fecha_Hora_Registro)

- DNI: PRIMARY KEY
- DNI CHECK (DNI ~ '\d{8}[A-Za-z]$')
- Nombre: NOT NULL
- Apellido_1: NOT NULL
- Fecha_Nacimiento: NOT NULL
- Fecha_Nacimiento BETWEEN 1900 AND FECHA_ACTUAL
- Nombre_Usuario: NOT NULL
- Nombre_Usuario: UNIQUE
- Sexo CHECK IN [Masculino, Femenino, Otros]
- Contraseña: NOT NULL
- Activo TRIGGER Activo = True: existe un periodo que no haya finalizado
- Fecha_Hora_Registro: NOT NULL
- Fecha_Hora_Registro TRIGGER Fecha_Hora_Registro = Fecha_Hora_Inserción

En la tabla 'trabajador' destaca la cantidad de **atributos no nulos** que esta presenta. Vemos también que el nombre de usuario es un atributo único, pues si nuestros trabajadores van a acceder a una aplicación deberán hacerlo a través de un determinado 'username' y contraseña. Pero quizás, de lo más importante de esta tabla, es el campo **'active'** cuyo valor varía a través de un disparador que comprueba los periodos en los que trabaja.

5. Prestación

Prestacion(ID, ID_Articulo, DNI_Trabajador, ID_Usuario_Adulto, ID_Usuario_Menor, PeriodoInicio, PeriodoFin, FechaDevolución, Vigente)

- ID: PRIMARY KEY
- ID_Articulo: FOREIGN KEY de Articulo(ID)
- DNI_Trabajador: FOREIGN KEY de Trabajador(DNI)
- ID_Usuario_Adulto: FOREIGN KEY de Usuario_Adulto(ID)
- ID_Usuario_Menor: FOREIGN KEY de Usuario_Menor(ID)
- PeriodoInicio TRIGGER PeriodoInicio = Fecha de inserción
- PeriodoFin TRIGGER PeriodoFin = Periodo Inicio + 14 dias
- Vigente TRIGGER Vigente = True IF FechaDevolución IS NOT NULL
- TRIGGER En caso de inserción: un usuario no puede tener más 5 prestaciones vigentes (o 7 si es estudiante o menor)
- TRIGGER (ID_Usuario_Adulto != NULL && ID_Usuario_Menor = NULL) || (ID_Usuario_Adulto = NULL && ID_Usuario_Menor != NULL)
- TRIGGER: Actualizar el stock de cada articulo con la creacion o finalizacion de cada prestacion

En la tabla 'prestacion' contamos con hasta 4 claves ajenas que hacen referencia a los artículos, trabajadores y usuarios. Esta es la tabla con más disparadores puesto que está **diseñada para que todas las fechas se generen automáticamente** al crear una entrada. Y además, por la restricción que comentamos anteriormente de la limitación del número de prestaciones.

En resumen, el modelo relacional difiere del modelo entidad relación en ser mucho más específico en cuanto a la definición de requisitos, valores y restricciones. Este ayudó a definir con mucha más precisión cada entidad y a tener en cuenta todos y cada uno de los 'trigger' que pueden ser disparados por las mismas.

Implementación en PostgreSQL

Tal y como se mencionó al principio del informe, el sistema gestor de base de datos utilizado para este proyecto es PostgreSQL. A través de este se ha creado una base de datos denominada "mylibrary", con todas sus relaciones.

Estructura de directorios y ficheros

Para la correcta implementación de la base de datos en PostgreSQL se han creado un total de 3 ficheros, además de un 4º adicional. Estos son:

- El archivo **initialize.sql**

  En este simplemente se elimina si existe y se crea la base de datos a utilizar para posteriormente conectarse a ella.

```
-- Dropping existing database
DROP DATABASE IF EXISTS mylibrary;

-- Creating database 'myLibrary'
CREATE DATABASE mylibrary;

-- Connecting to created database
\connect mylibrary
```

- El archivo **creations.sql**

  En este archivo se crea la estructura de cada una de las tablas a introducir. Aquí también podemos encontrar la definición de todas las funciones y disparadores creados.

```
/**
 * * * TABLE CREATIONS
 */

-- Tabla de articulos
CREATE TABLE articulo (
  id SERIAL PRIMARY KEY,
  tipo VARCHAR(19) CHECK (tipo in ('libro', 'materialPeriodico',
'materialAudiovisual')),
  titulo VARCHAR(50) NOT NULL,
  subtitulo VARCHAR(100),
  fchPublicacion DATE CHECK (fchPublicacion <=
CURRENT_DATE),
  portada BYTEA,
  stock INT NOT NULL CHECK (stock >= 0),
  disponible BOOLEAN GENERATED ALWAYS AS (stock > 0)
STORED
);
.
.
.
```

- El archivo **insertions.sql**

En este, se crean las inserciones para cada una de las tablas. Inserciones correctas, las cuales no deberían proporcionar fallo alguno.

```
-- Inserciones para la tabla 'articulo'
INSERT INTO articulo (titulo, tipo, subtitulo, fchPublicacion,
portada, stock)
VALUES
  ('Historia del Arte Moderno','libro','Una exploración de las
corrientes artísticas del siglo XX', '2022-10-15',
   E'\\x89504e470d0a1a0a0000',
   100),
  ('Recetas de Cocina Tradicional','libro', 'Platos caseros y
deliciosos', '2021-08-25',
   E'\\x89504e470d0a1a0a0000',
   100),
  ('Viaje al Centro de la Tierra','libro', 'Una emocionante novela
de aventuras', '2020-05-12',
   E'\\x89504e470d0a1a0a0000',
   100),
.
.
.
```

- El archivo **conflitCases.sql**

  Este último archivo es un fichero en el que se presentan toda la serie de errores que pueden generarse al interferir con algun valor fuera de dominio en alguno de los atributos, algún disparador o alguna función asociada a alguna de las tablas.

```
-- Tabla articulo
-- Stock Negativo
INSERT INTO articulo (titulo, tipo, subtitulo, fchPublicacion,
portada, stock)
VALUES
  ('Articulo_1','libro','Subtitulo_1', '2022-10-01',
E'\\x89504e470d0a1a0a0000', -1);

-- Tipo de Articulo erroneo
```

```
INSERT INTO articulo (titulo, tipo, subtitulo, fchPublicacion,
portada, stock)
VALUES
  ('Articulo_1','OtroTipo','Subtitulo_1', '2022-10-01',
E'\\x89504e470d0a1a0a0000', 1);

-- Fecha de publicacion de articulo mayor que fecha actual
INSERT INTO articulo (titulo, tipo, subtitulo, fchPublicacion,
portada, stock)
VALUES
  ('Articulo_1','libro','Subtitulo_1', '2099-10-01',
E'\\x89504e470d0a1a0a0000', 1);
.
.
.
```

Visualización de las tablas a través de terminal

La estructura de esta base de datos se aprecia a continuación bajo la siguiente imagen:

```
mylibrary=# \dt
                List of relations
 Schema |           Name            | Type  |  Owner
--------+---------------------------+-------+----------
 public | articulo                  | table | postgres
 public | autor                     | table | postgres
 public | autor_materialaudiovisual | table | postgres
 public | dia                       | table | postgres
 public | direccion                 | table | postgres
 public | emailtrabajador           | table | postgres
 public | emailusuario              | table | postgres
 public | generoarticulo            | table | postgres
 public | horario                   | table | postgres
 public | isla                      | table | postgres
 public | libro                     | table | postgres
 public | materialaudiovisual       | table | postgres
 public | materialperiodico         | table | postgres
 public | periodo                   | table | postgres
 public | prestacion                | table | postgres
 public | provincia                 | table | postgres
 public | tarjetasocio              | table | postgres
 public | telefonotrabajador        | table | postgres
 public | telefonousuario           | table | postgres
 public | trabajador                | table | postgres
 public | tutorusuariomenor         | table | postgres
 public | usuarioadulto             | table | postgres
 public | usuariomenor              | table | postgres
(23 rows)
```

Podemos apreciar que tenemos un conjunto de 23 relaciones o tablas. Un conjunto que hemos comentado repetidamente a lo largo del informe.

A continuación, pueden apreciarse algunas de las tablas perfectamente implementadas y rellenadas en la base de datos a partir de la ejecución de los anteriores ficheros. Todas estas imágenes se pueden encontrar en la carpeta '[media]' del repositorio.

- Para la tabla **autor**

```
mylibrary=# select * from autor;
 id | nombre | apellido1 | apellido2 | fchnacimiento | fchmuerte  | sexo | edad
----+--------+-----------+-----------+---------------+------------+------+------
  1 | Juan   | Pérez     | González  | 1980-05-15    |            | M    |   43
  2 | María  | López     | Martínez  | 1992-11-28    |            | F    |   31
  3 | Carlos | Ruiz      | Fernández | 1975-07-10    | 2022-06-30 | M    |   48
  4 | Ana    | García    | Sánchez   | 1988-03-20    |            | F    |   35
  5 | Pedro  | Rodríguez | Vega      | 1960-09-05    | 2018-12-18 | M    |   63
  6 | Isabel | Fernández | Hernández | 1978-12-12    |            | F    |   45
  7 | Manuel | Martínez  | Díaz      | 1985-06-03    |            | M    |   38
  8 | Laura  | Gómez     | Jiménez   | 1915-02-17    | 2001-12-18 | F    |  108
  9 | Javier | Serrano   | Muñoz     | 1972-08-22    |            | M    |   51
 10 | Elena  | Moreno    | Alonso    | 1983-04-08    |            | F    |   40
(10 rows)
```

- Para la tabla **materialPeriodico**

```
mylibrary=# select * from materialperiodico ;
 idarticulo | editorial  | numpaginas |   tipo
------------+------------+------------+-----------
         11 | Editorial3 |         20 | Periodico
         12 | Editorial4 |         15 | Periodico
         13 | Editorial6 |         15 | Periodico
         14 | Editorial7 |         20 | Periodico
         15 | Editorial8 |         25 | Periodico
         16 | Editorial8 |         25 | Revista
         17 | Editorial8 |         25 | Revista
         18 | Editorial8 |         25 | Revista
         19 | Editorial8 |         25 | Revista
         20 | Editorial9 |         18 | Revista
(10 rows)
```

- Para la tabla **email-trabajador**

```
mylibrary=# select * from emailtrabajador;
 dnitrabajador |           email
---------------+---------------------------
 12345678A     | juan.perez@example.com
 98765432B     | maria.gonzalez@example.com
 45678901C     | pedro.lopez@example.com
 78901234D     | ana.ruiz@example.com
 23456789E     | luis.fernandez@example.com
 34567890F     | elena.diaz@example.com
 89012345G     | carlos.jimenez@example.com
 01234567H     | laura.vega@example.com
 56789012I     | javier.morales@example.com
 67890123J     | sara.perez@example.com
(10 rows)
```

- Para la tabla **isla**

```
 mylibrary=# select * from isla;
  id | nombreisla |     latitud      |     longitud     | idprovincia
 ----+------------+------------------+------------------+-------------
   1 | Isla1      | (28.123,-16.987) | (28.123,-16.987) |           1
   2 | Isla2      | (27.456,-15.789) | (27.456,-15.789) |           2
   3 | Isla3      | (26.789,-14.567) | (26.789,-14.567) |           3
   4 | Isla4      | (25.012,-13.345) | (25.012,-13.345) |           4
   5 | Isla5      | (24.345,-12.123) | (24.345,-12.123) |           5
   6 | Isla6      | (23.678,-10.901) | (23.678,-10.901) |           6
   7 | Isla7      | (22.901,-9.789)  | (22.901,-9.789)  |           7
   8 | Isla8      | (22.234,-8.567)  | (22.234,-8.567)  |           8
   9 | Isla9      | (21.567,-7.345)  | (21.567,-7.345)  |           9
  10 | Isla10     | (20.89,-6.123)   | (20.89,-6.123)   |          10
 (10 rows)
```

Visualización de las casos conflictivos a través de terminal

De igual manera, se muestran algunas imágenes en las que se reflejan cada uno de los errores generados a partir de no respetar determinados disparadores, checks o restricciones de las tablas. Estas también se pueden encontrar en su totalidad en el directorio '[media]'.

- Para la tabla **artículo**
  - No puede haber Stock negativo

```
mylibrary=# INSERT INTO articulo (titulo, tipo, subtitulo, fchPublicacion,
 portada, stock)
VALUES
  ('Articulo_1','libro','Subtitulo_1', '2022-10-01', E'\\x89504e470d0a1a0a
0000', -1);
ERROR:  el nuevo registro para la relación «articulo» viola la restricción
 «check» «articulo_stock_check»
DETALLE:  La fila que falla contiene (27, libro, Articulo_1, Subtitulo_1,
2022-10-01, \x89504e470d0a1a0a0000, -1, f).
mylibrary=#
```

- Para la tabla **generoArticulo**
  - Evitar la duplicidad de los datos

```
mylibrary=# INSERT INTO generoArticulo (idArticulo, genero)
VALUES
  (1, 'Historia'),
  (1, 'Historia');
ERROR:  llave duplicada viola restricción de unicidad «generoarticulo_pkey
»
DETALLE:  Ya existe la llave (idarticulo, genero)=(1, Historia).
mylibrary=#
```

- Para la tabla **libro**
  - Insertar un artículo que no sea de tipo libro en la tabla libro

```
mylibrary=# INSERT INTO libro (idArticulo, editorial, numPaginas, estilo, s
nopsis, idAutor)
VALUES
  (11, 'Editorial1', 300, 'Lirico', 'Recopilación de noticias ', 1);
ERROR:  el articulo a insertar en la tabla 'libro' no posee el tipo 'libro'
CONTEXTO:  función PL/pgSQL check_tipo_libro() en la línea 4 en RAISE
mylibrary=#
```

API realizada en Flask

La parte que queda por comentar de todo el proyecto es el despliegue de la base de datos a través del framework Flask. Esto es la creación de la API correspondiente.

Estructura de ficheros y directorios

Como se podrá ver en la carpeta denominada ‘apiFlask’ del repositorio del proyecto tenemos un archivo denominado “App.py”, que representa el archivo principal de ejecución de nuestro programa, seguido de otros archivos que representan el modelado de cada uno de los objetos proporcionados por la API.

- El archivo **App.py**

  Contiene la conexión con la base de datos (para la cual deberán cambiarse los parámetros en el caso de un usuario o de otro) y cada una de las rutas con sus respectivos metodos (GET, POST, PATCH o DELETE) implementadas.

- El **resto** de archivos

  Contienen la definición de cada una de las funciones empleadas en cada ruta para cada uno de los objetos mapeados. Estos son 'articulo', 'libro', 'usuario', 'trabajador', etc.

Tablas de rutas

Las distintas tablas de rutas con sus respectivas definiciones pueden apreciarse a continuación.

- **Autores**

Métodos	URLs	Acciones
GET	/mylibrary/autores	Devuelve todos los autores
GET	/mylibrary/autores/id/:id	Devuelve un autor dada su ID
POST	/mylibrary/autores	Crea un nuevo autor
PATCH	/mylibrary/autores/id/:id	Edita un autor dada su ID
DELETE	/mylibrary/autores/id/:id	Elimina un autor dada su ID

- **Artículos**

Métodos	URLs	Acciones
GET	/mylibrary/articulos	Devuelve todos los artículos
GET	/mylibrary/articulos/id/:id	Devuelve un articulo dada su ID
GET	/mylibrary/articulos/titulo/:title	Devuelve un articulo dado su titulo
DELETE	/mylibrary/articulos/id/:id	Elimina un artículo dada su ID

- **Libros**

Métodos	URLs	Acciones
GET	/mylibrary/articulos/libros	Devuelve todos los libros
GET	/mylibrary/articulos/libros/id/:id	Devuelve un libro dada su ID
GET	/mylibrary/articulos/libros/titulo/:title	Devuelve un libro dado su título
GET	/mylibrary/articulos/libros/autor/id/:id	Devuelve todos los libros de un autor dada la ID del autor
POST	/mylibrary/articulos/libros	Crea un libro
PATCH	/mylibrary/articulos/libros/id/:id	Edita un libro dada su ID
DELETE	/mylibrary/articulos/llibros/id/:id	Elimina un libro dada su ID

- **Trabajadores**

  Por motivos de seguridad, se ha decidido no implementar la devolución de las contraseñas de los trabajadores a la hora de realizar las peticiones.

Métodos	URLs	Acciones
GET	/mylibrary/trabajadores	Devuelve todos los trabajadores
GET	/mylibrary/trabajadores/dni/:dni	Devuelve un trabajador dado su DNI

- **Usuarios Adultos**

Métodos	URLs	Acciones
GET	/mylibrary/usuarioAdulto	Devuelve todos los usuarios adultos
GET	/mylibrary/usuarioAdulto/id/:id	Devuelve un usuario adulto mediante su ID
GET	/mylibrary/usuarioAdulto/dni/:dni	Devuelve un usuario adulto

		mediante el ID de su tarjeta de socio
POST	/mylibrary/usuarioAdulto	Crea un usuario adulto
PATCH	/mylibrary/usuarioAdulto/id/:id	Actualiza un usuario adulto
DELETE	/mylibrary/usuarioAdulto/id/:id	Borra un usuario por su ID
DELETE	/mylibrary/usuarioMenor/dni/:dni	Borra un usuario por su dni

- **Usuarios Menores**

Métodos	URLs	Acciones
GET	/mylibrary/usuarioMenor	Devuelve todos los usuarios menores
GET	/mylibrary/usuarioMenor/id/:id	Devuelve un usuario menor mediante su ID
GET	/mylibrary/usuarioMenor/id_TarjetaSocio/:idTarjeta	Devuelve un usuario menor mediante el ID de su tarjeta de socio
POST	/mylibrary/usuarioMenor	Crea un usuario menor
PATCH	/mylibrary/usuarioMenor/id/:id	Actualiza un usuario menor
DELETE	/mylibrary/usuarioMenor/id/:id	Borra un usuario por su ID
DELETE	/mylibrary/usuarioMenor/id_TarjetaSocio/:idTarjeta	Borra un usuario por la ID de su Tarjeta de Socio

- **Prestaciones**

Para crear una prestación únicamente son necesarios los campos 'idArticulo', 'dniTrabajador' e 'idUsuario'. Por otro lado, actualizar una prestación simplemente permite actualizar su fecha de devolución.

Métodos	URLs	Acciones
GET	/mylibrary/prestaciones	Devuelve todas las prestaciones
GET	/mylibrary/prestaciones/id/:id	Devuelve una

		prestación dada su ID
GET	/mylibrary/prestaciones/articulo/id/:id	Devuelve todas las prestaciones asociadas a un artículo dada la ID del artículo
GET	/mylibrary/prestaciones/trabajador/dni/:dni	Devuelve todas las prestaciones asociadas a un trabajador dado el DNI del artículo
POST	/mylibrary/prestaciones	Crea una prestación
PATCH	/mylibrary/prestaciones/id/:id	Edita la fecha de devolución de una prestación dada su ID

Repositorio

El repositorio empleado para la realización de este proyecto se puede ubicar en el siguiente enlace.

Referencias

- 2023-2024 ADBD. Proyecto de la asignatura.pdf
- Red BICA | Biblioteca de Canarias (modelo de referencia)
- PostgreSQL
- Flask
- draw.io