



Documentación de Proyecto Final

Fundamentos de Bases de Datos



Elaborado por:

Miriam Fernanda Arellanes Pérez 353256

José Eduardo Conde Hernández 299506

Jesús Manuel Calleros Vázquez 348737

Grupo: 6CC2

Docente: José Saul De Lira Miramontes

Índice:

Manual Técnico	3
1. Sistema/Aplicación	3
a. Nombre del sistema	3
b. Descripción y delimitación del sistema	3
c. Objetivo general	3
d. Objetivos específicos	3
e. Descripción de tipos de usuarios	3
f. Entorno operativo del sistema	4
2. Especificación de requerimientos	4
a. Requerimientos funcionales	4
b. Modelado del sistema	5
i. Diagrama de clases	5
ii. Diagrama de casos de uso	6
iii. Diagrama de actividades	7
c. Requerimientos no funcionales	8
3. Arquitectura del Sistema/Aplicación.	8
a. Layer/Tier's	8
b. Frontback/Backend	9
c. Estructura modular del sistema	9
4. Base de datos	10
a. Diagrama Conceptual (Entidad/Relación)	10
b. Esquema Lógico de la Base de Datos	11
i. Especificación Tablas (Normalizadas hasta BCNF)	12
ii. Integridad de Datos (Constraints)	18
5. Lógica/Reglas del Negocio	20
6. Descripción Interfaz de la aplicación	21
7. Descripción de reglas de seguridad(acceso/operación)	23
8. Acceso a la Base de Datos mediante la Aplicación	23
9. Conclusión	31
10. Anexos	32
11. Bibliografía/Referencias	35

Manual Técnico

1. Sistema/Aplicación

a. Nombre del sistema

Sistema de Gestión bibliotecario Web

b. Descripción y delimitación del sistema

Es un sistema encargado de la administración y gestión de información relacionada con los libros, revistas, artículos y otros materiales de la biblioteca, así como información sobre los usuarios, préstamos, reservas y otros aspectos de la gestión de la biblioteca.

Este sistema podría incluir tablas para almacenar información sobre los libros, como el título, autor, editorial, año de publicación, número de ejemplares, ubicación, categoría, etc. También se puede contar con tablas para almacenar información sobre los usuarios de la biblioteca, como su nombre, dirección, número de identificación, historial de préstamos, multas, etc.

Además, la base de datos podría incluir tablas para almacenar información sobre los préstamos, reservas y devoluciones de los libros, incluyendo la fecha de préstamo, fecha de devolución, nombre del usuario, título del libro, etc. También podría haber tablas para almacenar información sobre las interacciones con los usuarios, como las solicitudes de información, sugerencias y quejas.

c. Objetivo general

Permitir la búsqueda y recuperación eficiente de información de la biblioteca, así como la gestión de préstamos, reservas y devoluciones de libros.

d. Objetivos específicos

- Permitir a los usuarios buscar y filtrar los libros ofrecidos por el sistema por género.
- Facilitar el proceso de renta de libros.
- Mantener un inventario actualizado de los libros disponibles para renta.
- De igual forma llevar un control de las multas por retardo al momento de entregar los libros.

e. Descripción de tipos de usuarios

- Público General Registrado
- Público General No Registrado

f. Entorno operativo del sistema

El sistema está diseñado para funcionar como una página web accesible desde los navegadores web comunes.

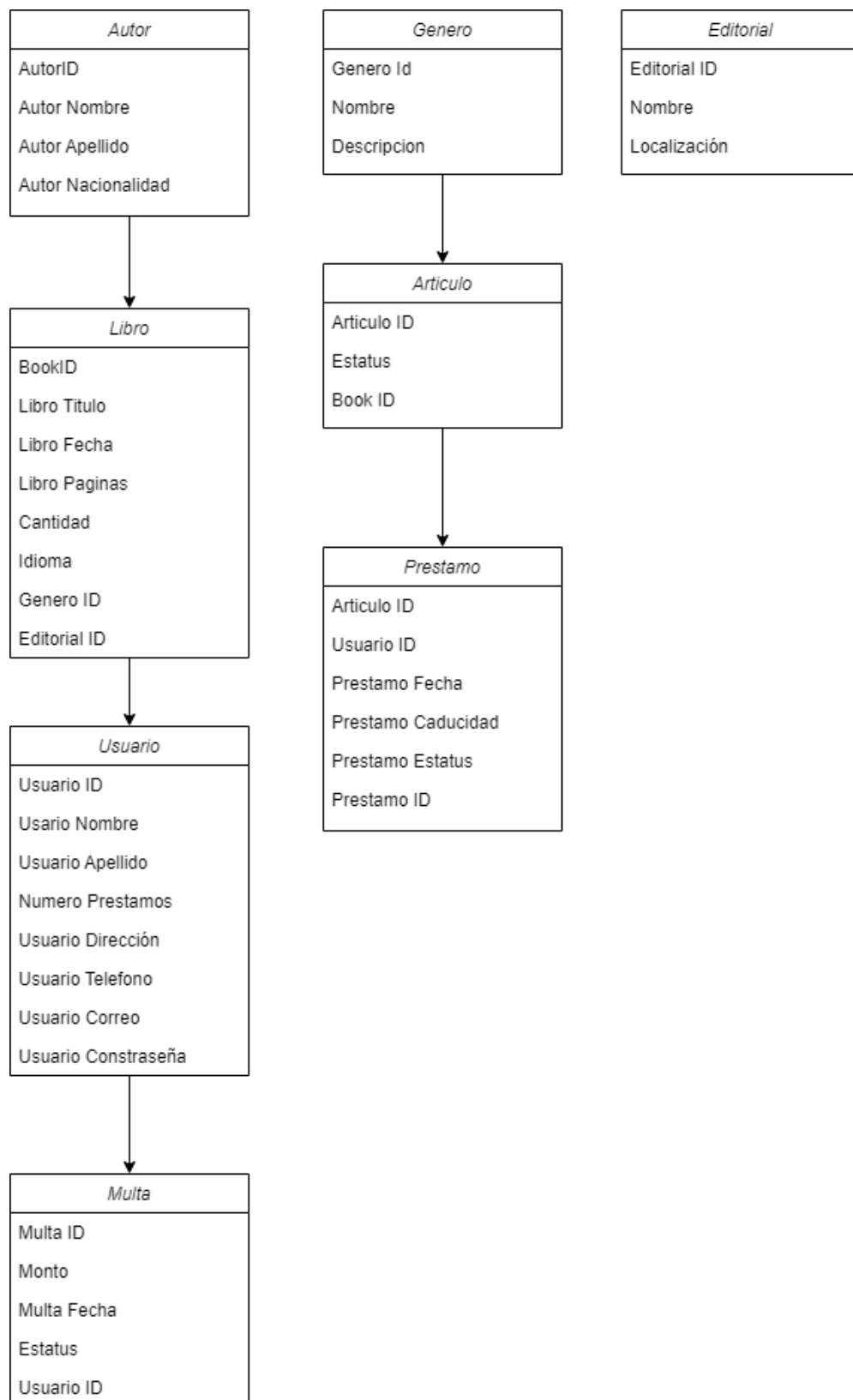
2. Especificación de requerimientos

a. Requerimientos funcionales

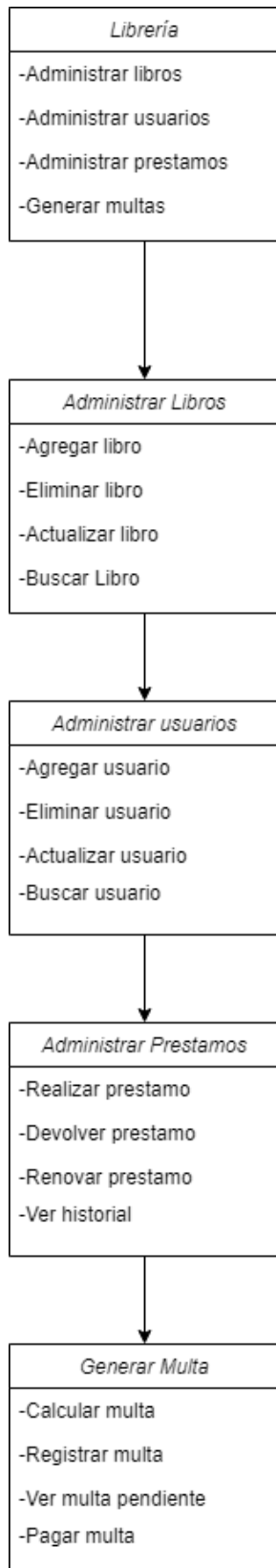
- El sistema debe permitir el registro de nuevos libros con todos los datos correspondientes.
- El sistema debe validar que el libro registrado no exista previamente en la base de datos.
- El sistema debe permitir a los usuarios registrarse en la página web proporcionando su nombre, apellido, dirección, número de teléfono, correo electrónico y contraseña.
- El sistema debe validar que el correo electrónico del usuario sea único.
- El sistema debe almacenar la información del usuario en la base de datos.
- El sistema debe permitir a los usuarios buscar libros por título o género.
- Los usuarios deben poder ver los resultados de búsqueda y seleccionar un libro para obtener más información.
- El sistema debe permitir a los usuarios alquilar libros disponibles.
- El sistema debe verificar la disponibilidad del libro y registrar el alquiler en la base de datos.
- El sistema debe mantener un registro de los libros alquilados por cada usuario.
- El sistema debe verificar que el libro se devuelva dentro del tiempo correspondiente y sin daños.
- En caso de retraso en la devolución, el sistema debe generar una multa y registrarla en la base de datos.
- El sistema debe permitir a los usuarios consultar el estado de sus alquileres y multas asociadas.
- Los usuarios deben poder ver una lista de los libros que tienen actualmente y cualquier multa pendiente.

b. Modelado del sistema

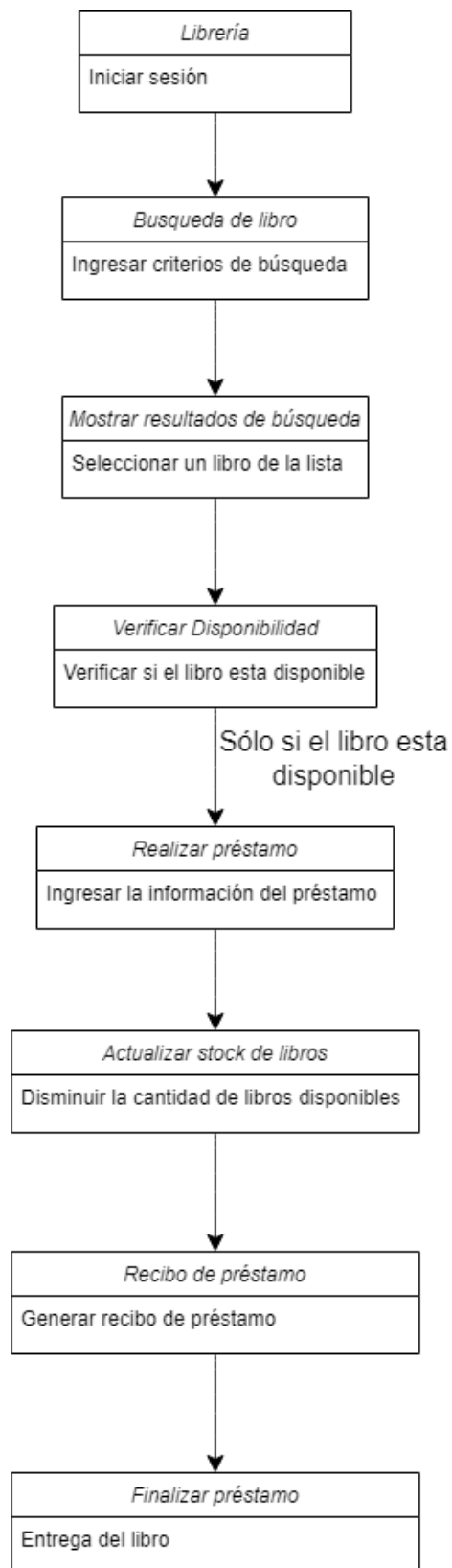
i. Diagrama de clases



ii. Diagrama de casos de uso



iii. Diagrama de actividades



c. Requerimientos no funcionales

- El sistema debe ser capaz de manejar grandes volúmenes de datos de forma eficiente y con tiempos de respuesta rápidos.
- El tiempo de respuesta para consultas y operaciones en la base de datos debe ser aceptable para garantizar una experiencia de usuario fluida.
- El sistema debe garantizar la seguridad de los datos almacenados en la base de datos, evitando el acceso no autorizado, la modificación o eliminación indebida de información.
- Debe implementarse un mecanismo de copias de seguridad y recuperación de la base de datos para garantizar la disponibilidad de los datos en caso de fallas.
- El sistema debe estar disponible las 24 horas del día, los 7 días de la semana, para que los usuarios puedan acceder a él en cualquier momento.
- Debe haber medidas en su lugar para minimizar el tiempo de inactividad planificado o no planificado, como mantenimiento programado y redundancia de servidores.
- El sistema debe ser confiable y estar disponible para su uso en todo momento, evitando interrupciones o fallas frecuentes.
- Debe tener una gestión adecuada de errores y excepciones para minimizar las interrupciones en el funcionamiento normal.

3. Arquitectura del Sistema/Aplicación.

a. Layer/Tier's

Capa de Presentación (Frontend): Esta capa se encarga de la interfaz de usuario y la interacción con el usuario. Está compuesta por tecnologías web como HTML, CSS y JavaScript, utilizando el framework Materialize CSS para el diseño visual y la interfaz responsiva. Aquí se encuentra la página de inicio, la búsqueda de libros, los detalles del libro, las órdenes y las multas.

Capa de Aplicación (Backend): Esta capa maneja la lógica de negocio y la funcionalidad de la aplicación. Utiliza Node.js como entorno de ejecución de JavaScript y Express como framework web para el enrutamiento y la gestión de las solicitudes HTTP. Se encarga de manejar las solicitudes del usuario, procesar la lógica de la aplicación y realizar operaciones en la base de datos.

Capa de Datos (Backend): Esta capa se conecta a la base de datos Oracle utilizando el paquete oracledb de Node.js. Se encarga de realizar consultas y actualizaciones en la base de datos, recuperar y almacenar la información relacionada con los usuarios, los libros, las órdenes y las multas.

b. Frontback/Backend

- El frontend de la aplicación utiliza tecnologías web como HTML, CSS y JavaScript para la interfaz de usuario y la interacción con el usuario, utilizamos el framework materialize de css.
- El backend de la aplicación utiliza Node.js, Express y OracleDB para manejar las solicitudes del usuario, procesar la lógica de la aplicación y realizar operaciones en la base de datos Oracle.

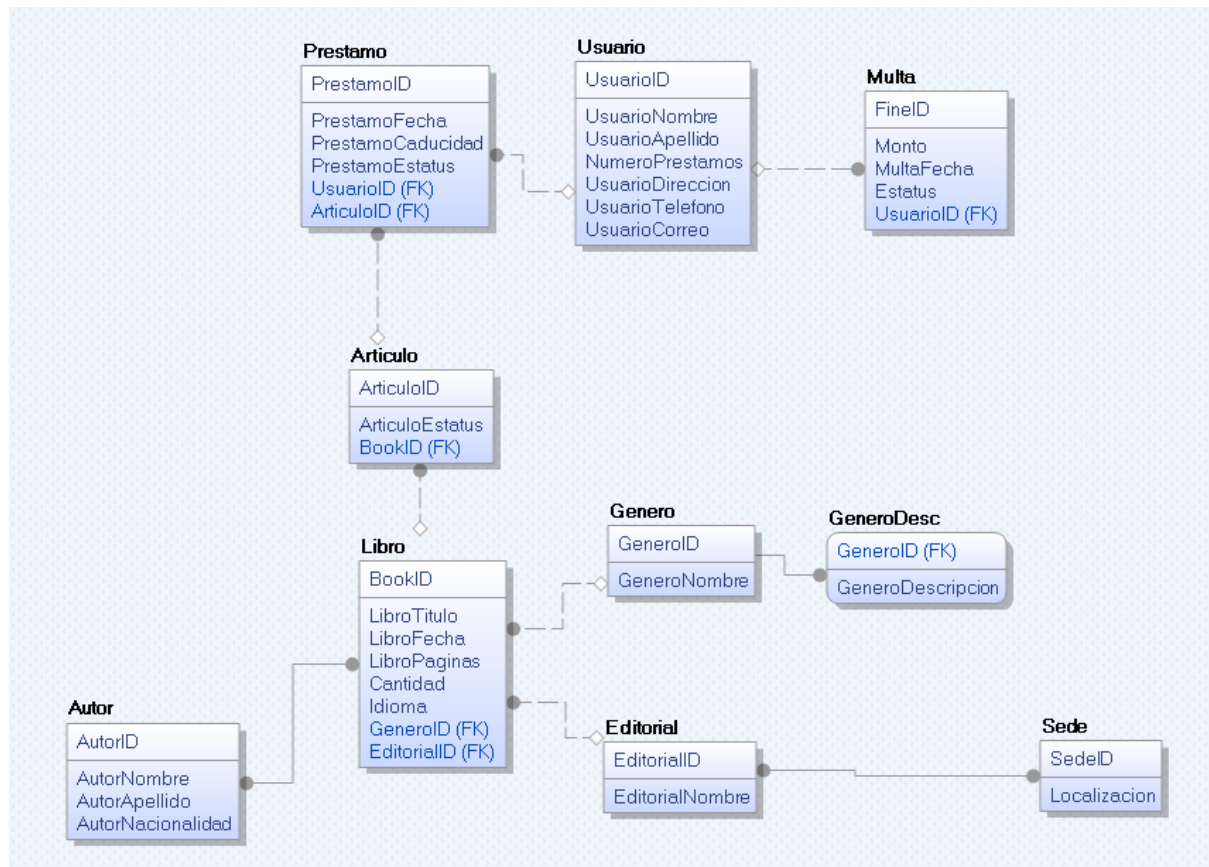
c. Estructura modular del sistema

La estructura modular del sistema sigue un enfoque MVC (Modelo-Vista-Controlador) o similar, donde los componentes del sistema están organizados en módulos y siguen un patrón de separación de responsabilidades. Algunos módulos son los siguientes:

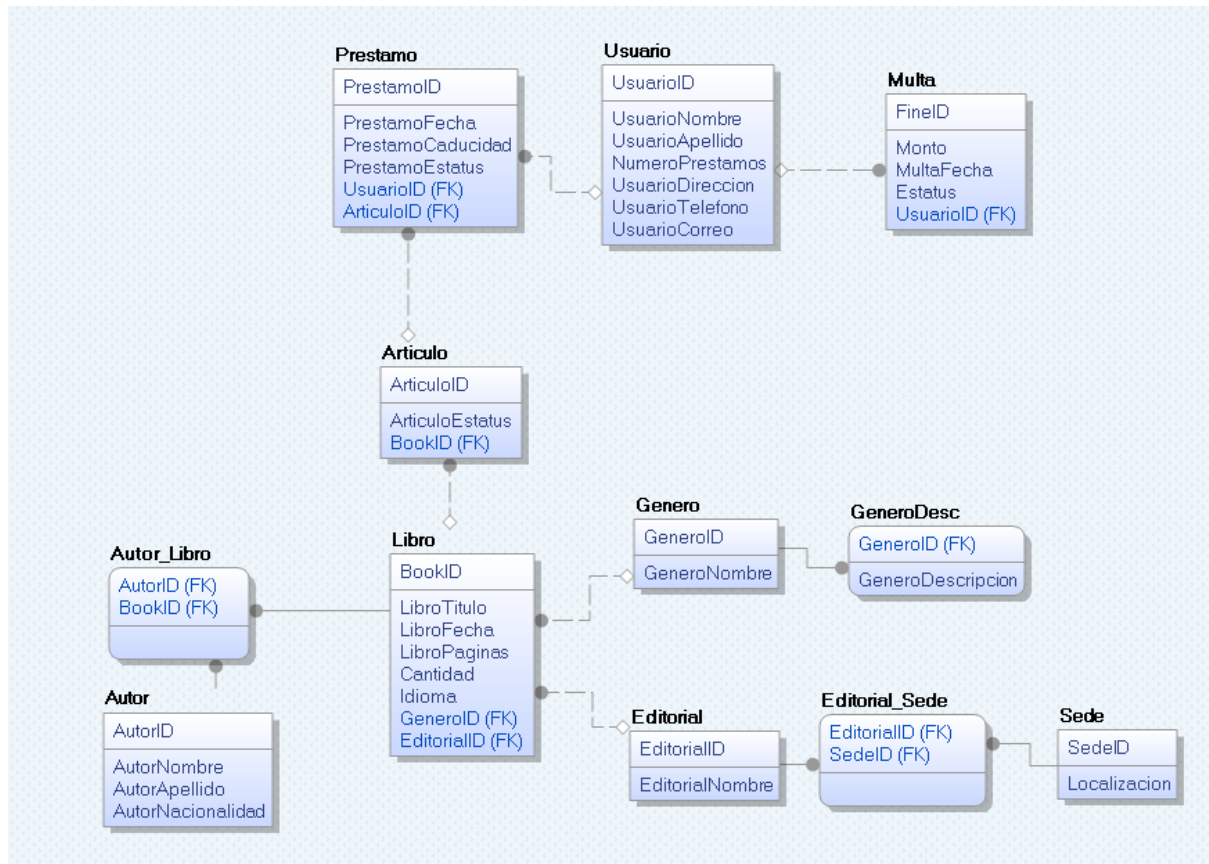
- Módulo de Autenticación: Maneja el registro de usuarios, el inicio de sesión y la gestión de sesiones.
- Módulo de Búsqueda y Filtrado: Permite a los usuarios buscar libros por nombre y filtrar por categorías.
- Módulo de Gestión de Libros: Maneja las operaciones relacionadas con los libros, como obtener detalles, actualizar el stock, etc.
- Módulo de Órdenes de Préstamo: Gestiona la generación de órdenes de préstamo, verificación de stock, fechas de préstamo y caducidad, etc.
- Módulo de Seguimiento de Órdenes: Permite a los usuarios ver el estado de sus préstamos y actualizar el estado de las órdenes (entregado, multa pendiente, etc.).
- Módulo de Multas: Maneja la generación y visualización de multas, así como su estado y pagos.

4. Base de datos

a. Diagrama Conceptual (Entidad/Relación)



b. Esquema Lógico de la Base de Datos



i. Especificación Tablas (Normalizadas hasta BCNF)

Tabla autor

Autor:

- AutorID (PK)
 - AutorNombre
 - AutorApellido
 - AutorNacionalidad
- $\{\text{AutorID}\} \rightarrow \{\text{AutorNombre}, \text{AutorApellido}, \text{AutorNacionalidad}\}$

1NF:

Una relación está en la primera forma normal (1FN) si y sólo si cada atributo tiene un valor sencillo para cada tupla. ✓

2NF:

Una relación está en segunda forma normal (2FN) si y sólo si está en primera forma normal y todos los atributos no clave son completamente dependientes funcionales sobre la clave. ✓

3NF:

Una relación está en tercera forma normal (3FN) si, siempre que exista una dependencia funcional no trivial $X \rightarrow A$, entonces o X es una superclave o A es un miembro de alguna clave candidata. ✓

BCNF:

Una relación está en forma Boyce/Codd (FNBC) si, siempre que existe una dependencia funcional no trivial $X \rightarrow A$, entonces X es una superclave. ✓

Tabla Genero

Genero:

- GeneroID (PK)
 - GeneroNombre (fk)
- $\{\text{GeneroID}\} \rightarrow \{\text{GeneroNombre}\}$

1NF:

Una relación está en la primera forma normal (1FN) si y sólo si cada atributo tiene un valor sencillo para cada tupla. ✓

2NF:

Una relación está en segunda forma normal (2FN) si y sólo si está en primera forma normal y todos los atributos no clave son completamente dependientes funcionales sobre la clave. ✓

3NF:

Una relación está en tercera forma normal (3FN) si, siempre que exista una dependencia funcional no trivial $X \rightarrow A$, entonces o X es una superclave o A es un miembro de alguna clave candidata. ✓

BCNF:

Una relación está en forma Boyce/Codd (FNBC) si, siempre que existe una dependencia funcional no trivial $X \rightarrow A$, entonces X es una super-

clave. ✓

Tabla **Genero**Descripcion

Genero:

-GeneroNombre (PK)

-GeneroDescripcion

{GeneroNombre} → { GeneroDescripcion }

1NF:

Una relación está en la primera forma normal (1FN) si y sólo si cada atributo tiene un valor sencillo para cada tupla. ✓

2NF:

Una relación está en segunda forma normal (2FN) si y sólo si está en primera forma normal y todos los atributos no clave son completamente dependientes funcionales sobre la clave. ✓

3NF:

Una relación está en tercera forma normal (3FN) si, siempre que exista una dependencia funcional no trivial $X \rightarrow A$, entonces o X es una superclave o A es un miembro de alguna clave candidata. ✓

BCNF:

Una relación está en forma Boyce/Codd (FNBC) si, siempre que existe una dependencia funcional no trivial $X \rightarrow A$, entonces X es una super-clave. ✓

Tabla **Editorial**:

EditorialID (PK)

EditorialNombre

{EditorialID} → { EditorialNombre }

1NF:

Una relación está en la primera forma normal (1FN) si y sólo si cada atributo tiene un valor sencillo para cada tupla. ✓

2NF:

Una relación está en segunda forma normal (2FN) si y sólo si está en primera forma normal y todos los atributos no clave son completamente dependientes funcionales sobre la clave. ✓

3NF:

Una relación está en tercera forma normal (3FN) si, siempre que exista una dependencia funcional no trivial $X \rightarrow A$, entonces o X es una superclave o A es un miembro de alguna clave candidata. ✓

BCNF:

Una relación está en forma Boyce/Codd (FNBC) si, siempre que existe una dependencia funcional no trivial $X \rightarrow A$, entonces X es una super-clave. ✓

Tabla Sede:

-SedeID (PK)
-Localizacion
 $\{SedeID\} \rightarrow \{Localizacion\}$

1NF:

Una relación está en la primera forma normal (1FN) si y sólo si cada atributo tiene un valor sencillo para cada tupla. ✓

2NF:

Una relación está en segunda forma normal (2FN) si y sólo si está en primera forma normal y todos los atributos no clave son completamente dependientes funcionales sobre la clave. ✓

3NF:

Una relación está en tercera forma normal (3FN) si, siempre que exista una dependencia funcional no trivial $X \rightarrow A$, entonces o X es una superclave o A es un miembro de alguna clave candidata. ✓

BCNF:

Una relación está en forma Boyce/Codd (FNBC) si, siempre que existe una dependencia funcional no trivial $X \rightarrow A$, entonces X es una superclave. ✓

Tabla Editorial_Sede:

-EditorialID (PK, FK hacia Editorial.EditorialID)
-SedeID (PK, FK hacia Sede.SedeID)
 $\{EditorialID, SedeID\} \rightarrow \{EditorialID, SedeID\}$

1NF:

Una relación está en la primera forma normal (1FN) si y sólo si cada atributo tiene un valor sencillo para cada tupla. ✓

2NF:

Una relación está en segunda forma normal (2FN) si y sólo si está en primera forma normal y todos los atributos no clave son completamente dependientes funcionales sobre la clave. ✓

3NF:

Una relación está en tercera forma normal (3FN) si, siempre que exista una dependencia funcional no trivial $X \rightarrow A$, entonces o X es una superclave o A es un miembro de alguna clave candidata. ✓

BCNF:

Una relación está en forma Boyce/Codd (FNBC) si, siempre que existe una dependencia funcional no trivial $X \rightarrow A$, entonces X es una superclave. ✓

Libro

- BookID (PK)
- LibroTitulo
- LibroFecha
- LibroPaginas
- Cantidad
- Idioma
- GeneroID(FK)
- EditorialID(FK)

$\{\text{BookID}\} \rightarrow \{\text{LibroTitulo}, \text{LibroFecha}, \text{LibroPaginas}, \text{cantidad}, \text{idioma}, \text{generoID}, \text{EditorialID}\}$

1NF:

Una relación está en la primera forma normal (1FN) si y sólo si cada atributo tiene un valor sencillo para cada tupla. ✓

2NF:

Una relación está en segunda forma normal (2FN) si y sólo si está en primera forma normal y todos los atributos no clave son completamente dependientes funcionales sobre la clave. ✓

3NF:

Una relación está en tercera forma normal (3FN) si, siempre que exista una dependencia funcional no trivial $X \rightarrow A$, entonces o X es una superclave o A es un miembro de alguna clave candidata. ✓

BCNF:

Una relación está en forma Boyce/Codd (FNBC) si, siempre que existe una dependencia funcional no trivial $X \rightarrow A$, entonces X es una superclave. ✓

Artículo

- ArticuloID (PK)
- ArticuloEstatus
- BookID(FK)

$\{\text{ArticuloID}\} \rightarrow \{\text{ArticuloEstatus}, \text{BookID}\}$

1NF:

Una relación está en la primera forma normal (1FN) si y sólo si cada atributo tiene un valor sencillo para cada tupla. ✓

2NF:

Una relación está en segunda forma normal (2FN) si y sólo si está en primera forma normal y todos los atributos no clave son completamente dependientes funcionales sobre la clave. ✓

3NF:

Una relación está en tercera forma normal (3FN) si, siempre que exista una dependencia funcional no trivial $X \rightarrow A$, entonces o X es una superclave o A es un miembro de alguna clave candidata. ✓

BCNF:

Una relación está en forma Boyce/Codd (FNBC) si, siempre que existe una dependencia funcional no trivial $X \rightarrow A$, entonces X es una superclave. ✓

Usuario:

- UsuarioID (PK)
- Usuarionombre
- Usuarioapellido
- Numeroprestamos
- Usuariodireccion
- Usuariotelefono
- Usuariocorreo
- Usuariocontraseña

$\{\text{UsuarioID}\} \rightarrow \{\text{UsuarioNombre, UsuarioApellido, numeroPrestamos, usuariodireccion, usuariotelefono, usuariocorreo, usuariocontraseña}\}$
 $\{\text{NumeroPrestamos}\} \rightarrow \{\text{usuarioID}\}$
 $\{\text{usuariocorreo}\} \rightarrow \{\text{UsuarioID, NumeroPrestamos}\}$

1NF:

Una relación está en la primera forma normal (1FN) si y sólo si cada atributo tiene un valor sencillo para cada tupla. ✓

2NF:

Una relación está en segunda forma normal (2FN) si y sólo si está en primera forma normal y todos los atributos no clave son completamente dependientes funcionales sobre la clave. ✓

3NF:

Una relación está en tercera forma normal (3FN) si, siempre que exista una dependencia funcional no trivial $X \rightarrow A$, entonces o X es una superclave o A es un miembro de alguna clave candidata. ✓

BCNF:


Una relación está en forma Boyce/Codd (FNBC) si, siempre que existe una dependencia funcional no trivial $X \rightarrow A$, entonces X es una superclave. ✓

Prestamo


- ArticuloID(FK)
- UsuarioID(FK)
- PrestamoFecha
- PrestamoCaducidad
- PrestamoEstatus
- PrestamoID (PK)

$\{\text{PrestamoID}\} \rightarrow \{\text{PrestamoFecha}, \text{PrestamoCaducidad}, \text{PrestamoEstatus}, \text{PrestamoID}, \text{ArticuloID}, \text{UsuarioID}\}$


1NF:

Una relación está en la primera forma normal (1FN) si y sólo si cada atributo tiene un valor sencillo para cada tupla. 


2NF:

Una relación está en segunda forma normal (2FN) si y sólo si está en primera forma normal y todos los atributos no clave son completamente dependientes funcionales sobre la clave. 

3NF:

Una relación está en tercera forma normal (3FN) si, siempre que exista una dependencia funcional no trivial $X \rightarrow A$, entonces o X es una superclave o A es un miembro de alguna clave candidata. 

BCNF:


Una relación está en forma Boyce/Codd (FNBC) si, siempre que existe una dependencia funcional no trivial $X \rightarrow A$, entonces X es una superclave. 

Multa:


- MultaID (PK)
- Monto
- Multafecha
- Estatus
- UsuarioID(FK)
- ArticuloID(FK)

$\{\text{MultaID}\} \rightarrow \{\text{multaFecha}, \text{monto}, \text{estatus}, \text{usuarioID}, \text{ArticuloID}\}$

1NF:

Una relación está en la primera forma normal (1FN) si y sólo si cada atributo tiene un valor sencillo para cada tupla. 

2NF:

Una relación está en segunda forma normal (2FN) si y sólo si está en primera forma normal y todos los atributos no clave son completamente dependientes funcionales sobre la clave. 

3NF:

Una relación está en tercera forma normal (3FN) si, siempre que exista una dependencia funcional no trivial $X \rightarrow A$, entonces o X es una superclave o A es un miembro de alguna clave candidata. ✓

BCNF:

Una relación está en forma Boyce/Codd (FNBC) si, siempre que existe una dependencia funcional no trivial $X \rightarrow A$, entonces X es una super-clave. ✓

ii. Integridad de Datos (Constraints)

```
ALTER TABLE Libro  
ADD CONSTRAINT XPKLibro PRIMARY KEY (LibroID);
```

```
ALTER TABLE Artículo  
ADD CONSTRAINT XPKArticulo PRIMARY KEY (ArticuloID);
```

```
ALTER TABLE Autor  
ADD CONSTRAINT XPKAutor PRIMARY KEY (AutorID);
```

```
ALTER TABLE Autor_Libro  
ADD CONSTRAINT XPKAutor_Libro PRIMARY KEY (AutorID, LibroID);
```

```
ALTER TABLE Editorial  
ADD CONSTRAINT XPKEditorial PRIMARY KEY (EditorialID);
```

```
ALTER TABLE Genero  
ADD CONSTRAINT XPKGenero PRIMARY KEY (GeneroID);
```

```
ALTER TABLE GeneroDesc  
ADD CONSTRAINT XPKGeneroDesc PRIMARY KEY (GeneroID);
```

```
ALTER TABLE Sede  
ADD CONSTRAINT XPKSede PRIMARY KEY (SedeID);
```

```
ALTER TABLE Editorial_Sede  
ADD CONSTRAINT XPKEditorial_Sede PRIMARY KEY (EditorialID,  
SedeID);
```

```
ALTER TABLE Usuario  
ADD CONSTRAINT XPKUsuario PRIMARY KEY (UsuarioID);
```

```
ALTER TABLE Multa  
ADD CONSTRAINT XPKMulta PRIMARY KEY (FineID);
```

Universidad Autónoma de Chihuahua
Facultad de Ingeniería

```
ALTER TABLE Prestamo
  ADD CONSTRAINT XPKPrestamo PRIMARY KEY (PrestamoID);

ALTER TABLE Libro
  ADD CONSTRAINT R_16 FOREIGN KEY (GeneroID) REFERENCES
  Genero (GeneroID) ON DELETE SET NULL;

ALTER TABLE Libro
  ADD CONSTRAINT R_21 FOREIGN KEY (EditorialID) REFERENCES
  Editorial (EditorialID) ON DELETE SET NULL;

ALTER TABLE Articulo
  ADD CONSTRAINT R_23 FOREIGN KEY (LibroID) REFERENCES Libro
  (LibroID) ON DELETE SET NULL;

ALTER TABLE Autor_Libro
  ADD CONSTRAINT R_19 FOREIGN KEY (AutorID) REFERENCES Autor
  (AutorID);

ALTER TABLE Autor_Libro
  ADD CONSTRAINT R_20 FOREIGN KEY (LibroID) REFERENCES Libro
  (LibroID);

ALTER TABLE GeneroDesc
  ADD CONSTRAINT R_27 FOREIGN KEY (GeneroID) REFERENCES
  Genero (GeneroID);

ALTER TABLE Editorial_Sede
  ADD CONSTRAINT R_29 FOREIGN KEY (EditorialID) REFERENCES
  Editorial (EditorialID);

ALTER TABLE Editorial_Sede
  ADD CONSTRAINT R_30 FOREIGN KEY (SedeID) REFERENCES Sede
  (SedeID);

ALTER TABLE Multa
  ADD CONSTRAINT R_17 FOREIGN KEY (UsuarioID) REFERENCES
  Usuario (UsuarioID) ON DELETE SET NULL;

ALTER TABLE Prestamo
  ADD CONSTRAINT R_32 FOREIGN KEY (UsuarioID) REFERENCES
  Usuario (UsuarioID) ON DELETE SET NULL;

ALTER TABLE Prestamo
  ADD CONSTRAINT R_33 FOREIGN KEY (ArticuloID) REFERENCES
  Articulo (ArticuloID) ON DELETE SET NULL;

ALTER TABLE Usuario
```

5. Lógica/Reglas del Negocio

Las reglas de negocio y la lógica de la aplicación de Biblioteca Online se centran en el manejo de los siguientes aspectos:

Registro de Usuarios:

- Los usuarios deben proporcionar información válida y completa al registrarse en la aplicación.
- Se verifica que la dirección de correo electrónico sea única y no esté registrada previamente.
- Las contraseñas se almacenan de forma segura utilizando técnicas de hash y sal para proteger la información del usuario.

Inicio de Sesión:

- Los usuarios deben proporcionar credenciales válidas (correo electrónico y contraseña) para iniciar sesión en la aplicación.
- Se verifica la validez de las credenciales ingresadas y se autentica al usuario mediante una comparación con los datos almacenados en la base de datos.

Búsqueda y Filtrado de Libros:

- Los usuarios pueden buscar libros por su nombre utilizando la barra de búsqueda.
- Los libros se filtran por categorías seleccionadas, mostrando solo aquellos que correspondan a la categoría elegida.
- Los resultados de búsqueda y filtrado se presentan al usuario de manera dinámica y actualizada en la interfaz.

Detalles del Libro:

- Al hacer clic en el botón "+" de un libro, se accede a los detalles específicos de ese libro.
- Se muestra al usuario información detallada sobre el libro, como portada, título, ISBN, autor, género, editorial y stock disponible.

Órdenes de Préstamo:

- Los usuarios pueden realizar órdenes de préstamo de libros disponibles.
- Se verifica la disponibilidad de stock del libro antes de generar una orden de préstamo.
- Se registra la información de la orden, incluyendo la fecha de préstamo y la fecha de caducidad.

- Se proporciona al usuario una notificación de confirmación o error dependiendo del resultado de la generación de la orden.

Seguimiento de Órdenes y Multas:

- Los usuarios pueden acceder a la página de órdenes para ver el estado de sus préstamos.
- Se muestra al usuario una lista de sus órdenes de préstamo, incluyendo detalles como la imagen del libro, el ISBN, la fecha de préstamo y la fecha de caducidad.
- Se permite al usuario actualizar el estado de una orden, marcándola como "entregada" cuando el libro se devuelve o indicando una "multa pendiente" si el libro no se devuelve a tiempo.
- Las multas se generan automáticamente si un libro no se devuelve antes de la fecha de caducidad.
- Los usuarios pueden acceder a la página de multas para ver las multas pendientes, incluyendo detalles como la fecha de creación, el monto a pagar y el estado de la multa.

Estas reglas de negocio y lógica de la aplicación garantizan que los usuarios puedan registrarse, buscar, ordenar y realizar un seguimiento de sus préstamos de libros de manera eficiente y precisa, siguiendo las normas y restricciones establecidas en el sistema de biblioteca.

6. Descripción Interfaz de la aplicación

La aplicación de Biblioteca FCM presenta una interfaz intuitiva y fácil de usar para los usuarios. A continuación, se proporciona una descripción detallada de los componentes y características clave de la interfaz desde una perspectiva técnica.

Página de Inicio:

La página de inicio muestra un formulario de registro para que los usuarios se registren en la aplicación. Está diseñada utilizando HTML y CSS (Materialize CSS) para lograr una apariencia atractiva y receptiva. El formulario de registro utiliza campos como nombre, dirección de correo electrónico, contraseña, etc., para recopilar la información del usuario. El registro se realiza a través de una solicitud POST a través de Node.js y Express.

Inicio de Sesión:

Después de registrarse, los usuarios pueden iniciar sesión en la aplicación utilizando su dirección de correo electrónico y contraseña. El inicio de sesión se realiza a través de un formulario de inicio de sesión en el panel izquierdo de la interfaz. Los

datos ingresados por el usuario se validan y se realiza una verificación con la base de datos de Oracle utilizando el módulo oracledb en Node.js.

Página Principal:

La página principal muestra una lista de libros registrados en la biblioteca. Cada libro se presenta con su imagen, título y un botón "+" para ordenar el libro. La lista de libros se obtiene de la base de datos de Oracle y se muestra dinámicamente en la interfaz utilizando plantillas HTML que nosotros diseñamos.

Barra Lateral:

La barra lateral ubicada en el lado izquierdo de la interfaz contiene tres botones: "Salir", "Multas" y "Órdenes". Estos botones están implementados como elementos HTML interactivos y se comunican con el servidor a través de solicitudes HTTP para realizar acciones correspondientes, como cerrar la sesión, acceder a la página de multas o acceder a la página de órdenes.

Barra de Búsqueda y Filtrado:

En la parte superior de la página principal, hay una barra de búsqueda que permite a los usuarios buscar libros por su nombre. Justo debajo de la barra de búsqueda, se encuentran los botones de filtrado por categorías. Al hacer clic en estos botones, se envían solicitudes al servidor para obtener libros de la categoría seleccionada y se actualiza la lista de libros en la interfaz utilizando

Detalles del Libro:

Cuando un usuario hace clic en el botón "+" de un libro, se le redirige a la página de detalles del libro. Aquí se muestran detalles más específicos sobre el libro, como la portada, título, ISBN, autor, género, editorial y stock disponible. Los detalles se obtienen de la base de datos de Oracle y se muestran en la interfaz.

Órdenes:

La página de órdenes muestra todas las órdenes de préstamo realizadas por el usuario. Para cada orden, se muestra la imagen del libro prestado, el ISBN, la fecha de préstamo y la fecha de caducidad. Además, se muestra un botón de estado que permite al usuario actualizar el estado de la orden. Los datos de las órdenes se obtienen de la base de datos de Oracle, específicamente de la tabla prestamos.

Multas:

En la página de multas, se muestran todas las multas pendientes del usuario. Cada multa incluye detalles como la fecha de creación, el monto a pagar y el estado de la multa. Los datos de las multas se obtienen de la base de datos de Oracle de la tabla multas.

7. Descripción de reglas de seguridad(acceso/operación)

Autenticación y Autorización:

- Se verifica la identidad del usuario utilizando credenciales únicas (correo electrónico y contraseña).
- Solo los usuarios autenticados tienen acceso a sus datos personales, así como de multas y órdenes

Protección de Contraseñas:

- Las contraseñas se transmiten de manera segura a través de conexiones cifradas para prevenir el acceso no autorizado durante la transmisión.

Gestión de Sesiones:

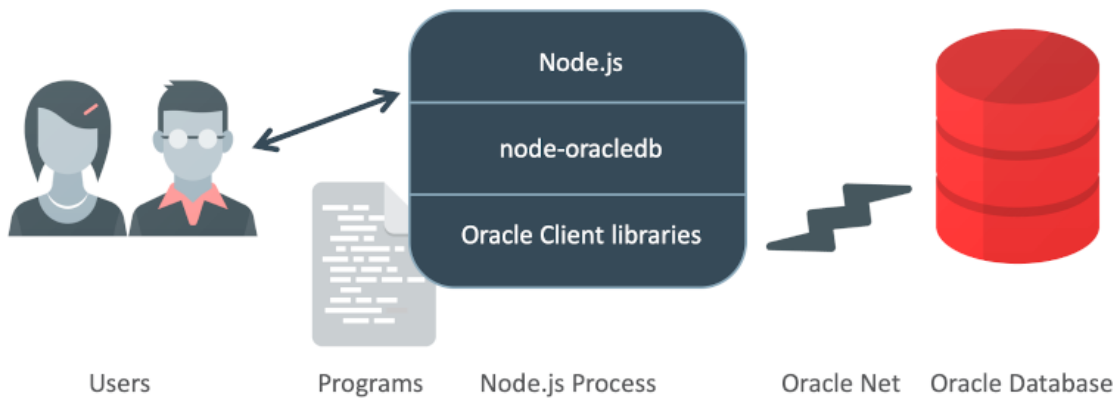
- Se utiliza un mecanismo de gestión de sesiones para mantener el estado del usuario.

8. Acceso a la Base de Datos mediante la Aplicación

Para acceder a la base de datos mediante nuestro código en Javascript de nuestro servidor usamos el módulo de Oracle DB.

OracleDB es un módulo de Node.js que proporciona una interfaz de programación de aplicaciones (API) para interactuar con una base de datos Oracle. Permite a los desarrolladores de Node.js conectarse a una base de datos Oracle, ejecutar consultas SQL, administrar transacciones y manejar resultados de forma eficiente.

Los usuarios interactúan con una aplicación Node.js, por ejemplo realizando peticiones web. El programa de la aplicación hace llamadas a funciones de node-oracledb. Internamente node-oracledb carga dinámicamente las librerías Oracle Client. Se establecen conexiones desde node-oracledb a la base de datos Oracle. Esto permite que la aplicación utilice SQL, PL/SQL y SODA.



En nuestro código de Javascript contamos con diferentes funciones para los POST y para los GET que hacen uso de esta tecnología y así acceder a nuestra base de datos en Oracle Express edition.

A continuación se muestran ejemplos:

En nuestro servidor App.js contamos con funciones como:

Importamos el módulo

```
const oracledb = require('oracledb');
```

```
const dbConfig = {
  user: ' ',
  password: ' ',
  connectString: 'localhost:1521/xepdb1'
};
```

Esta función, la usamos para guardar los datos de acceso a nuestra base de datos local, se usa como parte de llamada a otras funciones para poder crear la conexión.

Universidad Autónoma de Chihuahua
Facultad de Ingeniería

```
app.post('/login', async (req, res) => {
  const { email, password } = req.body;

  try {
    const connection = await oracledb.getConnection(dbConfig);
    const result = await connection.execute(
      `SELECT * FROM usuario WHERE usuariocorreo = :email AND usuariocontrasena = :password`,
      [email, password]
    );

    console.log(result.rows);

    const successAlert = `
    <script>
      alert('Usuario logeado exitosamente');
      window.location.href = '/books.html'; // Redirige a otra página después de mostrar el alert
    </script>
    `;

    await connection.close();

    if (result.rows.length > 0) {
      // Se encontró una coincidencia en la base de datos
      loggedUserID = result.rows[0][0];

      res.send(successAlert);
    } else {
      // No se encontraron coincidencias en la base de datos
      res.status(401).json({ message: 'Credenciales inválidas' });
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Error al comparar los datos con la base de datos' });
  }
});
```

Como se puede observar en esta función, es un POST de html, en esta se tiene una constante connection en la cual se almacena la conexión con la base de datos usando la constante anterior dbconfig. Después se tiene otra constante llamada result en la cual almacenamos los datos de una consulta de sql usando la constante anterior de connection; en este caso la consulta es para hacer un login, donde buscamos de la tabla usuario todas las coincidencias cuando el usuariocorreo sea tal y el usuariocontraseña sea tal.

Si se tienen coincidencias se puede ingresar a la página principal.

```
app.post('/reg', async (req, res) => {  
  try {  
    let usuarioid = generateUserID();  
  
    const {  
      usuarionombre,  
      usuarioapellido,  
      usuariodireccion,  
      usuariotelefono,  
      usuariocorreos,  
      usuarioccontrasena  
    } = req.body;  
  
    const connection = await oracledb.getConnection(dbConfig);  
    const result = await connection.execute(  
      `INSERT INTO usuario (usuarioid, usuarionombre, usuarioapellido, usuariodireccion, usuariotelefono, usuariocorreos, usuarioccontrasena)  
      VALUES (:usuarioid, :usuarionombre, :usuarioapellido, :usuariodireccion, :usuariotelefono, :usuariocorreos, :usuarioccontrasena)`,  
      [usuarioid, usuarionombre, usuarioapellido, usuariodireccion, usuariotelefono, usuariocorreos, usuarioccontrasena]  
    );  
  
    await connection.commit();  
    await connection.close();  
  
    const alertHTML = `  
    <script>  
      alert('Usuario registrado exitosamente, por favor inicie sesión');  
      window.location.href = '/index.html'; // Redirige a otra página después de mostrar el alert  
    </script>  
    `;  
    res.send(alertHTML);  
  } catch (error) {  
    console.log(error);  
    res.status(500).json({ error: 'Error al registrar el usuario' });  
  }  
});
```

En el siguiente ejemplo tenemos otro tipo de POST, el cual usamos como registro, lo que hacemos en este caso es recopilar la información de un formulario en nuestro documento html y pasarlo a la función, esta procesa los datos individualmente y los almacena en variables. Una vez más llamamos a la conexión mediante:

await oracledb.getConnection(dbconfig)

Después realizamos la consulta de sql al tener la conexión establecida con la base de datos. En esta consulta lo que queremos es insertar los datos del formulario para almacenarlos en la tabla usuario y así contar con registro del usuario que quiere interactuar con nuestra plataforma. Lo hacemos mediante:

*Insert into usuario(atributos de la tabla) values (nombres para las variables)
[variables creadas en la función]*

Por último se confirman los cambios en la base de datos con:

await connection.commit();

y cerramos la conexión:

await connection.close();

Ahora para el uso de los GET

```
app.get('/user0', async (req, res) => {
  await checkLoanStatus();
  let connection;
  const userID = loggedUserID;
  console.log(userID);

  try {
    connection = await oracledb.getConnection(dbConfig);

    const query = `SELECT DISTINCT p.PrestamoID, p.PrestamoFecha, p.PrestamoCaducidad, p.PrestamoEstatus,
    const result = await connection.execute(query, [userID]);
    // Transforma los resultados de la consulta en un formato adecuado
    const datos = result.rows.map(row => ({
      loanID: row[0],
      date: row[1],
      expiredDate: row[2],
      status: row[3],
      userID: row[4],
      articleID: row[5],
      title: decodeURIComponent((row[6])),
      isbn: row[7]
    }));

    res.json(datos);
  } catch (error) {
    console.error('Error al obtener datos desde OracleDB:', error);
    res.status(500).json({ error: 'Error al obtener datos' });
  } finally {
    if (connection) {
      try {
        await connection.close();
      } catch (error) {
        console.error('Error al cerrar la conexión:', error);
      }
    }
  }
});
```

En esta función hacemos el manejo de un GET que utilizaremos para mostrar los préstamos o pedidos de libros de un usuario.

Creamos la conexión con:

```
await oracledb.getConnection(dbconfig);
```

Creamos la consulta que vamos a realizar a la base de datos, en este caso, queremos tomar todo de la tabla préstamos, pero a su vez también queremos tener el nombre del libro específico del préstamo, puesto que nuestra tabla préstamos no cuenta con ese atributo si no con la llave foránea articuloID, la cual la usamos para encontrar el libroNombre en la tabla articulo. La consulta es de la siguiente manera:

```
SELECT DISTINCT p.PrestamoID, p.PrestamoFecha,
p.PrestamoCaducidad, p.PrestamoEstatus, p.UsuarioID, p.ArticuloID,
l.LibroTitulo AS LibroNombre
FROM Prestamo p
LEFT JOIN Articulo a ON p.ArticuloID = a.ArticuloID
LEFT JOIN Libro l ON a.LibroID = l.LibroID;
```

Seguido a esto almacenamos las coincidencias en una variables y la procesamos para separar los datos por atributo en variables individuales, estas variables serán unidas en un archivo json que será leído por el html correspondiente para dar formato a los datos y mostrarlos.

Estos son los dos tipos principales de manejo del acceso a la base de datos por parte de nuestro servidor.

Contamos con más ejemplos que tratan a sus funciones, la conexión y las consultas de la misma forma o alguna variante ligera.

```
async function checkLoanStatus() {
  try {
    const connection = await oracledb.getConnection(dbConfig);

    // Obtener los préstamos del usuario logeado
    const userLoans = await connection.execute(
      'SELECT * FROM Prestamo WHERE UsuarioID = :userID',
      [loggedUserID]
    );

    for (const loan of userLoans.rows) {
      const loanID = loan[0];
      const expiredDate = new Date(loan[2]);
      const status = loan[3];
      console.log(expiredDate)

      // Verificar si el préstamo está vencido
      if (expiredDate < new Date() && status === 'Activo') {
        // Actualizar el estatus del préstamo a 'multa pendiente'
        await connection.execute(
          'UPDATE Prestamo SET PrestamoEstatus = :status WHERE PrestamoID = :loanID',
          ['multa pendiente', loanID]
        );

        // Crear una multa para el usuario
        const fineID = generateLoanId();
        const fineAmount = 100; // Monto de la multa

        const fineStatus = 'Pendiente';

        await connection.execute(
          `INSERT INTO Multa (MultaID, Monto, MultaFecha, estatus, UsuarioID)
          VALUES (:fineID, :fineAmount, SYSDATE, :fineStatus, :userID)`,
          [fineID, fineAmount, fineStatus, loggedUserID]
        );
      }
    }
    await connection.commit();
    await connection.close();
  } catch (error) {
    console.error(error);
  }
}
```

```
app.post('/order', async (req, res) => {
  prestamoID = generateLoanId();
  const { isbn } = req.body;
  const userID = loggedUserID;
  console.log(userID, isbn);
  const successAlert = `
    <script>
      alert('Usuario logeado exitosamente');// Redirige a otra página después de mostrar el alert
    </script>
  `;
  try {
    const connection = await oracledb.getConnection(dbConfig);
    const result = await connection.execute(
      `SELECT a.ArticuloID
      FROM Articulo a
      LEFT JOIN Prestamo p ON a.ArticuloID = p.ArticuloID
      WHERE a.LibroID = :isbn AND p.ArticuloID IS NULL AND ROWNUM = 1
      `,
      [isbn]
    );
    await connection.close();

    if(result.rows[0] == undefined){
      throw new Error("Ya no hay stock disponible");
    }
    else{
      articuloID = result.rows[0][0];
      const connection = await oracledb.getConnection(dbConfig);
      const loan = await connection.execute(
        `INSERT INTO Prestamo (PrestamoID, PrestamoFecha, PrestamoCaducidad, PrestamoEstatus, UsuarioID, ArticuloID)
        VALUES (:prestamoid, SYSDATE, SYSDATE + INTERVAL '5' MINUTE, 'Activo', :userID, :articuloID)`,
        [prestamoID, userID, articuloID]
      );
      const minusStock = await connection.execute(
        `UPDATE Libro SET Cantidad = Cantidad - 1 WHERE LibroID = :isbn`,
        [isbn]
      );
      await connection.commit();
      await connection.close();
      console.log("Prestamo registrado exitosamente");
      res.sendStatus(200);
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Error al registrar el prestamo' });
  }
});
```

9. Conclusión

En resumen este proyecto realizado para la clase de Introducción a las bases de datos ha logrado que podamos conectar con el servidor que elaboramos en JavaScript, creando así un sitio web el cual es de una biblioteca en la cual cumplomos el prestar libros a los usuarios registrados y de igual manera la generación de multas para préstamos que se atrasaron.

Durante el proceso de desarrollo se puso un énfasis en la experiencia del usuario y que la interfaz presentada fuera sencilla y amigable con el usuario.

La integración con la base de datos permite un almacenamiento eficiente de la información, asegurando que a futuro se pueda consultar la información.

En conclusión el proyecto ha logrado satisfacer las necesidades y requerimientos establecidos, junto con ello logramos aprender a manejar un sitio web y el cómo poder relacionarlo con la materia, logrando así el proyecto que estamos presentando.


10. Anexos

Página Log In o Sign In

¿Estas registrado?

Si

Por favor ingresa con tu cuenta




E-mail _____

Contraseña _____

INGRESAR

No

Por favor regístrate



Nombres _____ Apellidos _____

Dirección _____

Teléfono _____

E-mail _____


Contraseña _____

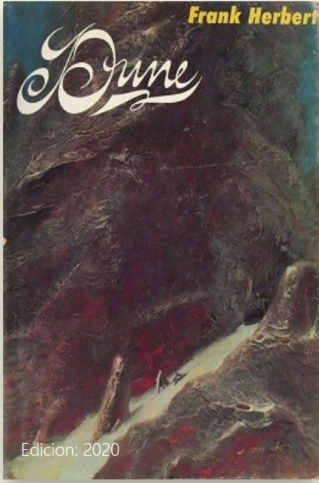
REGISTRARSE

Página principal



Página de libro específico





Edición: 2020

Dune

ISBN: 9782724215908

Autor: Frank Herbert

Páginas: 784

Idioma: Español


Género: Ciencia ficción

Editorial: Penguin Random House

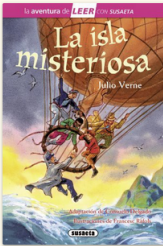
En stock: 16

ORDENAR

Página de órdenes



IR A MULTAS



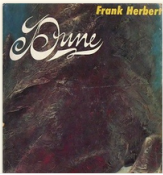
La isla misteriosa

Id de préstamo: 7951260427

Pedido realizado el: 2023-05-22T20:37:30.000Z

Fecha de devolución límite: 2023-05-29T20:37:30.000Z

ACTIVO



Dune

Id de préstamo: 4480034863

Pedido realizado el: 2023-05-22T19:08:52.000Z

Fecha de devolución límite: 2023-05-29T19:08:52.000Z

Página de multas



Mis multas


Favor de pasar a pagar sus multas en sucursal

[IR A ORDENES](#)

Id de multa : 1
Monto a pagar \$: 100
Fecha de multa : 2023-05-22T06:00:00.000Z
Usuario ID : 193845

PENDIENTE

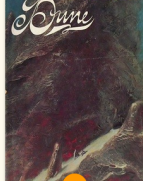
Sección de filtro



Search
dune

Categorías

- ☒ Todos
- ☐ Arte
- ☐ Aventura
- ☐ Ciencia ficción
- ☐ Fantasía
- ☐ Filosofía
- ☐ Historia
- ☐ Terror



Dune

11. Bibliografía/Referencias

Documentation - Materialize. (2014). Materializecss.com.

<https://materializecss.com/>

Express - Node.js web application framework. (2017). Expressjs.com.

<https://expressjs.com/>

OpenLibrary.org. (2023). Bienvenido a la Biblioteca Abierta | Biblioteca Abierta. Openlibrary.org. <https://openlibrary.org/>

"oracledb". NPM. Recuperado el 11 de agosto de 2021, de

<https://www.npmjs.com/package/oracledb>