

**TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA**



INGENIERÍA BIOMÉDICA



**TECNOLÓGICO
NACIONAL DE MÉXICO**

Asignatura:

Base de datos

Docente:

Ramirez Arzate Fortunato

Actividad:

Examen unidad 3

Grupo:

BM 5-B

Alumnos:

Armenta Medina Osmar Ediel
Gonzalez Olguin Martin Alonso
Lopez Gastelum Alan
Perez Garcia Diego
Vazquez Gomez Javier

Tijuana, Baja California 27 de Diciembre del 2025

ÍNDICE.....	1
Introducción.....	2
Requisitos técnicos mínimos.....	4
1. Instalación de dependencias.....	4
2. Estructura del proyecto.....	4
Estructura principal con:.....	4
3. nodemon.json.....	5
4. Variables de entorno (.env y .env.example).....	5
Explicación de la Base de Datos.....	6
1. Creación de la base de datos.....	6
2. Tabla usuarios.....	6
3. Tabla instrumentos.....	7
4. Tabla opcional préstamos.....	7
5. Creación del usuario no-root.....	7
Funcionalidad requerida.....	8
1. Login/Logout con bcrypt.....	8
2. Roles del sistema.....	8
3. CRUD de instrumentos.....	8
4. Búsqueda en vivo.....	9
5. Excel (subir y descargar).....	9
Conclusión General.....	10

Introducción

En este documento se hará un resumen sobre lo elaborado en el Examen de la unidad 3 “mini hakaton” en el cual hicimos un sitio web para el préstamo de instrumentación de laboratorio. En equipo implementamos funciones clave como autenticación, roles, CRUD de instrumentos, búsqueda en vivo y manejo de datos en Excel, apoyándose en herramientas como .env, nodemon, Bootstrap, Live Share y GitHub. El objetivo es demostrar dominio técnico y trabajo colaborativo en un entorno similar al desarrollo real.

Objetivo del examen: Construir un sistema web sencillo que permita gestionar el catálogo de instrumentos y registrar préstamos básicos. Debe incluir login, roles, CRUD de instrumentos, búsqueda en vivo y carga/descarga de datos en Excel.

Organización

El primer paso al reunirnos fue el organizarnos y dar roles, el encargado de generar el servidor fue alan y los roles del examen fueron los siguientes:

Bloque	Responsable
Live Share + GitHub	Alan
BD + usuario no-root	Alan
Login/Logout (bcrypt)	Martin
Roles (Admin/Asistente/Auditor)	Javier
CRUD instrumentos	Osmar
Búsqueda en vivo	Martin
Excel (subir/bajar)	Diego
UI con Bootstrap	Javier

Requisitos técnicos mínimos

1. Instalación de dependencias

Se utilizaron las dependencias necesarias para levantar el servidor, manejar la base de datos, proteger contraseñas, gestionar archivos Excel y cargar variables de entorno:

- express: para crear el servidor y las rutas.
- mysql2: para conectarse a la base de datos.
- dotenv: para leer variables desde el archivo .env.
- bcrypt: para encriptar contraseñas.
- multer: para recibir archivos Excel desde formularios.
- xlsx: para leer y generar archivos Excel.
- nodemon (desarrollo): reiniciar el servidor automáticamente al detectar cambios.

Se instalaron con:

```
npm i express mysql2 dotenv bcrypt multer xlsx  
npm i -D nodemon
```

2. Estructura del proyecto

Se organizó el proyecto siguiendo una estructura sencilla pero funcional para mantener el código ordenado:

/mini-hackaton-lab

Estructura principal con:

- server.js: archivo principal del servidor, donde se configuraron Express, rutas, conexión a BD y middlewares.
- package.json: contiene dependencias y scripts.
- nodemon.json: indica qué archivos observar y cómo ejecutar el servidor.
- .env: variables sensibles (no se sube a GitHub).
- .env.example: plantilla de las variables para que otros desarrolladores sepan qué configurar.

/db/schema.sql

Contiene las instrucciones para crear tablas como usuarios, instrumentos, y opcionalmente préstamos

Sirvió para inicializar la base de datos.

/uploads

Carpeta temporal donde Multer guarda los archivos Excel subidos antes de procesarlos.

/public

Carpeta estática para HTML, CSS y JS del cliente:

- index.html: página principal.
- login.html: formulario de inicio de sesión.
- instrumentos.html: CRUD de instrumentos.
- prestamos.html: gestión de préstamos (opcional).
- busqueda.html: página con búsqueda en vivo.
- navbar.html: menú reutilizable.
- styles.css: estilos personalizados.

3. nodemon.json

Se configuró un archivo simple y eficiente para desarrollo:

```
{  
  "watch": ["server.js", "public"],  
  "exec": "node server.js"  
}
```

Con esto, nodemon reinicia el servidor automáticamente cuando cambia el servidor o los archivos públicos, permitiendo avanzar más rápido.

4. Variables de entorno (.env y .env.example)

Se definieron variables mínimas para que el servidor se conectara a la base de datos y estableciera el puerto de ejecución:

```
PORT=3000  
DB_HOST=localhost  
DB_USER=lab_user  
DB_PASS=lab_pass
```

DB_NAME=laboratorio

- .env: contiene datos reales; se ignora en Git para proteger contraseñas.
- .env.example: contiene datos ficticios y se sube a Git como guía para que otros configuren su propio archivo.

Explicación de la Base de Datos

1. Creación de la base de datos

El script inicia creando la base de datos llamada laboratorio, usando utf8mb4 para soportar acentos, símbolos y caracteres modernos. Luego selecciona esa base para trabajar:

```
CREATE DATABASE IF NOT EXISTS laboratorio CHARACTER SET utf8mb4;  
USE laboratorio;
```

Esto garantiza que la BD exista sin duplicarse si el script se ejecuta varias veces.

2. Tabla usuarios

Se creó una tabla para almacenar la información de los usuarios del sistema:

- id: identificador único.
- nombre: nombre del usuario.
- correo: único, para evitar duplicados.
- password_hash: contraseña cifrada con bcrypt.
- rol: define permisos en el sistema (ADMIN, ASISTENTE, AUDITOR).

```
CREATE TABLE IF NOT EXISTS usuarios (...);
```

La tabla usa un ENUM para roles, lo que simplifica el control de acceso.

3. Tabla instrumentos

Esta tabla almacena los instrumentos del laboratorio, con información básica para el CRUD:

- nombre: nombre del instrumento.
- categoria: tipo/clase del equipo.
- estado: disponible, prestado o en mantenimiento.

- ubicacion: lugar donde se encuentra el instrumento.

CREATE TABLE IF NOT EXISTS instrumentos (...);

El estado por defecto es DISPONIBLE, ideal para nuevos registros.

4. Tabla opcional préstamos

Esta tabla permite llevar un registro mínimo de préstamos de instrumentos:

- instrumento_id: referencia al instrumento prestado.
- usuario_correo: quién lo solicitó.
- fecha_salida: se genera automáticamente.
- fecha_regreso: se actualiza cuando el instrumento se devuelve.

CREATE TABLE IF NOT EXISTS prestamos (...);

Incluye una clave foránea para asegurar que solo se registren préstamos de instrumentos existentes.

5. Creación del usuario no-root

Por seguridad, el sistema no se conecta con el usuario root de MySQL.

Se creó un nuevo usuario con su propia contraseña:

```
CREATE USER IF NOT EXISTS 'lab_user'@'localhost' IDENTIFIED BY 'lab_pass';
```

Luego se le otorgaron permisos solo sobre la base de datos laboratorio, evitando accesos innecesarios:

```
GRANT ALL PRIVILEGES ON laboratorio.* TO 'lab_user'@'localhost';  
FLUSH PRIVILEGES;
```

Esto permite que la aplicación acceda a las tablas sin exponer el usuario root y sigue buenas prácticas de seguridad.

Funcionalidad requerida

1. Login/Logout con bcrypt

El sistema incluye autenticación segura:

- Al registrarse, la contraseña del usuario se encripta con bcrypt y se guarda en password_hash.
- Al iniciar sesión, bcrypt compara la contraseña ingresada con el hash almacenado.
- El servidor mantiene la sesión del usuario (o token simple).
- Logout elimina o limpia esa sesión.

Esto garantiza que nunca se almacenen contraseñas en texto plano.

2. Roles del sistema

Se implementaron tres niveles de permisos:

ADMIN

- Puede ver, crear, editar y eliminar usuarios.
- Puede ver, crear, editar y eliminar instrumentos.

ASISTENTE

- Puede ver instrumentos.
- Puede editar o actualizar instrumentos.
- No puede borrar instrumentos ni administrar usuarios.

AUDITOR

- Solo puede ver la información.
- No puede modificar nada.

El middleware del servidor restringe rutas según el rol del usuario logueado.

3. CRUD de instrumentos

Se desarrollaron las operaciones esenciales para administrar el inventario:

- Crear: agregar instrumentos con nombre, categoría, estado y ubicación.
- Leer: listar todos los instrumentos o buscar uno específico.

- Actualizar: editar información de instrumentos existentes.
- Eliminar: borrar instrumentos cuando ya no son necesarios.

La información se almacena en la tabla instrumentos.

4. Búsqueda en vivo

Se agrego un campo de texto que busca en tiempo real:

- Cada vez que el usuario escribe, el frontend envía una petición a:

`/api/instrumentos?q=texto`

- El servidor responde con una lista filtrada.
- La tabla se actualiza sin recargar la página.

Esto mejora rapidez y usabilidad.

5. Excel (subir y descargar)

Subir .xlsx

- El usuario carga un archivo Excel con columnas: nombre, categoria, estado, ubicacion.
- Multer guarda el archivo temporalmente.
- La librería xlsx lo procesa.
- El servidor inserta o actualiza cada instrumento.

Esto permite cargar grandes cantidades de datos rápidamente.

Descargar

- El servidor genera un archivo instrumentos.xlsx con el catálogo actual.
- Se usa la librería xlsx para crearlo dinámicamente.
- El usuario lo descarga desde:

`GET /api/instrumentos/download`

Conclusión General

El desarrollo del Sistema Web para el Préstamo de Instrumentos de Laboratorio permitió integrar de manera práctica los conocimientos abordados en la Unidad 3, combinando fundamentos de bases de datos, seguridad, manejo de archivos y construcción de interfaces web. A través de un enfoque colaborativo, se implementaron funciones esenciales como la autenticación con bcrypt, control de roles, CRUD completo de instrumentos, búsqueda en vivo y carga/descarga mediante archivos Excel. Asimismo, se aplicaron buenas prácticas profesionales, como el uso de variables de entorno, un usuario no-root en la base de datos, nodemon para el entorno de desarrollo y Bootstrap para una interfaz limpia y funcional. En conjunto, esta mini hackatón simuló un entorno de trabajo real, fortaleciendo la capacidad del equipo para resolver problemas, organizar su código y construir soluciones eficientes en un tiempo limitado.