Trabajo Práctico N.º 3

Crear un nuevo repositorio en su cuenta de GitHub, en el cual se incluirá el backend y frontend del ejercicio que fue asignado aleatoriamente. Organizarlos en carpetas separadas (backend y frontend) dentro de la rama ‘master’. Solo se evaluarán los commits realizados dentro del período establecido entre la fecha de apertura y la fecha de cierre del trabajo práctico.

Requerimientos generales

- Frontend:
  - Framework: Vite + React (JavaScript).
  - Formulario de inicio de sesión y registro de usuario.
  - Comunicación con el backend mediante fetch consumiendo una API REST JSON.
  - Validaciones en el frontend (campos obligatorios, formato de datos, etc.).
- Backend:
  - Framework: Express.js.
  - Base de datos: MySQL (con conexión mediante mysql2).
  - Validaciones de datos con express-validator en todas las rutas de obtención, creación y actualización.
  - Manejo de errores con respuestas HTTP adecuadas (400, 401, 403, 404, 500).
  - Incluir archivos ‘http’ para prueba de la API.
  - Incluir archivo sql con las definiciones de las tablas.
- Autenticación y autorización:
  - Registro e inicio de sesión de usuarios.
  - Autenticación mediante JWT (JSON Web Tokens).
  - Middleware de Passport para verificar el token y restringir acceso a rutas protegidas.
  - Solo los usuarios autenticados pueden acceder a las operaciones CRUD.
- Seguridad:
  - Las contraseñas deben almacenarse encriptadas con bcrypt.
  - No deben enviarse contraseñas en texto plano.
  - Los tokens JWT deben expirar a las 4 horas.

Ejercicio A – Gestión de alumno, materias y notas

Desarrollar una aplicación web completa que permita gestionar alumnos, materias y sus calificaciones. A parte de los requisitos generales el ejercicio debe cumplir lo siguiente:

- Frontend:
  - Pantalla principal con paginas de:
    - Listado de alumnos.
    - Listado de materias.
    - Carga y visualización de notas por alumno y materia (hasta 3 notas).
  - Formularios para alta, modificación y eliminación de alumnos y materias.
  - Posibilidad de asignar notas a cada alumno en cada materia.
  - Mostrar el promedio de notas por materia y alumno.

- Backend:
  - Estructura de entidades:
    - Usuario: id, nombre, email, contraseña (encriptada con bcrypt).
    - Alumno: id, nombre, apellido, DNI (único).
    - Materia: id, nombre, código, año.
    - Nota: id, alumno_id, materia_id, nota1, nota2, nota3.

Ejercicio B – Gestión de pacientes, médicos y turnos

Desarrollar una aplicación que permita gestionar la asignación de turnos entre pacientes y médicos. A parte de los requisitos generales el ejercicio debe cumplir lo siguiente:

- Frontend:
  - Pantalla principal con paginas de:
    - Listado de pacientes.
    - Listado de médicos.
    - Carga y visualización de turnos por paciente y médico con modificación de estados.
  - Formularios para alta, modificación y eliminación de pacientes y médicos.
  - Posibilidad de asignar y administrar turnos.
  - Cambiar el estado del turno y registrar observaciones.
- Backend:
  - Estructura de entidades:
    - Usuario: id, nombre, email, contraseña (encriptada con bcrypt).
    - Paciente: id, nombre, apellido, DNI, fecha de nacimiento, obra social.
    - Médico: id, nombre, apellido, especialidad, matrícula profesional.
    - Turno: id, paciente_id, medico_id, fecha, hora, estado (pendiente, atendido, cancelado), observaciones.

Ejercicio C – Gestión de vehículos, conductores y viajes

Desarrollar una aplicación que permita registrar los vehículos de una empresa de transporte, sus conductores y los viajes realizados. A parte de los requisitos generales el ejercicio debe cumplir lo siguiente:

- Frontend:
  - Pantalla principal con paginas de:
    - Listado de vehículos.
    - Listado de conductores.
    - Carga y visualización de viajes por vehículos y conductores.
  - Formularios para alta, modificación y eliminación de vehículos y conductores.
  - Registro de viajes, asociando un vehículo y un conductor.
  - Consulta de historial de viajes por vehículo o por conductor.
  - Cálculo total de kilómetros recorridos por conductor o vehículo.
- Backend:
  - Estructura de entidades:
    - Usuario: id, nombre, email, contraseña (encriptada con bcrypt).
    - Vehículo: id, marca, modelo, patente, año, capacidad de carga.
    - Conductor: id, nombre, apellido, DNI, licencia, fecha de vencimiento de licencia.

- Viaje: id, vehiculo_id, conductor_id, fecha_salida, fecha_llegada, origen, destino, kilómetros, observaciones.

Criterios de evaluación

Criterio	Descripción	Valor (puntos)
Estructura del proyecto	Separación clara entre frontend y backend	10
Validación de datos	Uso correcto de express-validator y control de errores	15
Autenticación y autorización	Implementación correcta con JWT y Passport	20
Persistencia de datos	Conexión funcional y relaciones correctas en MySQL	15
Interfaz de usuario	Navegabilidad, claridad y validaciones en React	20
Seguridad	Encriptación de contraseñas, manejo de tokens	10
Commits del repositorio	Mensajes descriptivos, commits frecuentes que reflejen el avance del trabajo, y estructura limpia del repositorio.	10