PROGRAMACIÓN WEB AVANZADA

Tarea Proyecto

Tema: Solución de un caso práctico como proyecto final, individual en cualquiera de las siguientes opciones, definiendo objetivos, límites y alcance con el docente de la asignatura:

Objetivo: Desarrollar las habilidades fundamentales necesarias para el diseño e implementación de sistemas web interactivos y funcionales utilizando una combinación de HTML, BS 5, CSS, JavaScript y PHP.

Actividades para el logro del objetivo

Consideraciones generales

El ejercicio a desarrollar el estudiante debe considerar, aplicar HTML, BS 5, CSS, Javascript, un gestor gráfico MySQL como phpmyadmin y PHP, en páginas web (utilizando JS a su gusto).
El ejercicio planteado dispone de Nombre del Proyecto, función general, roles de usuario y base de datos, lo que constituye un punto de partida para desarrollar la programación.
El estudiante a su libre criterio, en el ejercicio que le corresponda estar asignado por sorteo, puede crear y/o modificar lo que considere pertinente a:

- El diseño de pantalla FrontEnd y BackEnd
- Las interacciones por rol de usuario
- Las funciones por rol de usuario
- La base de datos sugerida como punto de partida, y sus relaciones

Se le solicita

- No aplicar ningún Framework PHP.
- Aplicar seudo framework o un framework bajo criterio propio.
- Si no desea considerar enfoque framework puede trabajar modular y funcionalmente el Proyecto, sin necesidad obligatoria de aplicar algún enfoque de patrón de desarrollo.
- Incluir comentarios dentro de cada Programa que detallen en su lectura el apoyo para cualquier programador,
- Entregar un archivo rar donde consten:

- Carpeta del Proyecto, donde en la raíz conste la base de datos MySQL del mismo
- Un archivo word:
    - Dirección URL GitHub del Proyecto
        - En la cuenta GitHub ubicar en la raíz del Proyecto la base de datos MySQL
    - Dirección URL del Hospedaje Web donde instaló el Proyecto

El ejercicio a desarrollar el estudiante en forma individual será asignado por sorteo al azar en clase, acorde a cualquiera de las opciones:

1. Sistema de administración de una biblioteca
2. Sistema de administración de una tienda online
3. Sistema de gestión de una escuela
4. Sistema de gestión de una clínica
5. Sistema de gestión de una empresa de transporte
6. Sistema de gestión para un restaurante
7. Sistema de citas para una peluquería
8. Plataforma de crowdfunding
9. Sistema de administración de un zoológico
10. Sistema de gestión de un gimnasio
11. Sistema de reserva de billetes de avión
12. Gestor de pedidos para una floristería online
13. Sistema de gestión de una inmobiliaria
14. Sistema de gestión de una granja
15. Sistema de administración de un museo
16. Sistema de gestión para una agencia de viajes
17. Sistema de administración para una empresa de limpieza
18. Plataforma de gestión para una ONG
19. Sistema de gestión para un salón de belleza
20. Sistema de administración de un SPA
21. Sistema de citas para una clínica dental
22. Sistema de reservas para un campo de golf
23. Sistema de administración para una boda
24. Sistema de gestión para un salón de eventos
25. Sistema de citas para un taller de reparación de automóviles
26. Sistema de administración de un club deportivo
27. Sistema de reservas para un estudio de fotografía
28. Sistema de gestión para una ferretería online
29. Sistema de administración de una guardería
30. Plataforma de gestión para un sitio de noticias
31. Sistema de reservas para una sala de cine
32. Sistema de gestión de una lavandería
33. Sistema de gestión de una panadería
34. Sistema de gestión de una pizzería
35. Sistema de administración para una cervecera artesanal
36. Plataforma de gestión para un podcast
37. Sistema de gestión de una heladería
38. Sistema de administración para un granjero de mariscos
39. Sistema de reservas para una peluquería canina

40. Sistema de gestión para una joyería
41. Sistema de gestión para un taller de bicicletas
42. Sistema de administración de un vivero de plantas
43. Sistema de reservas para un parque de atracciones
44. Sistema de gestión para una empresa de reciclaje
45. Sistema de administración de un acuario
46. Plataforma para un servicio de entregas de comida a domicilio
47. Sistema de administración para un estudio de grabación musical

Ejercicio 1

Proyecto: Sistema de administración de una biblioteca

Función general: Este proyecto consiste en desarrollar un sistema de administración de una biblioteca orientado a la web. Los usuarios pueden registrarse en el sistema bajo varios roles, y su interacción con el sistema varían dependiendo del rol asignado.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar libros, autores, categorías y usuarios, y supervisar todas las operaciones del sistema.
- Bibliotecario: Puede añadir, editar y eliminar préstamos, devoluciones y renovaciones de libros, y gestionar los datos de los usuarios.
- Usuario: Puede buscar libros, consultar el estado de los préstamos y renovar los préstamos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Bibliotecario'), ('Usuario');

CREATE TABLE usuarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contrasena VARCHAR(100) NOT NULL,
  id_rol INT NOT NULL,
  FOREIGN KEY (id_rol) REFERENCES roles(id)
);

CREATE TABLE libros (
  id INT AUTO_INCREMENT PRIMARY KEY,
  titulo VARCHAR(100) NOT NULL,
  autor VARCHAR(100) NOT NULL,
  categoria VARCHAR(20) NOT NULL,
  esta_disponible BOOLEAN NOT NULL DEFAULT TRUE
);

CREATE TABLE prestamos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_usuario INT NOT NULL,
  id_libro INT NOT NULL,
  fecha_prestamo DATE NOT NULL,
  fecha_devolucion DATE NOT NULL,
  fecha_regreso DATE
);
```

Ejercicio 2

Proyecto: Sistema de administración de una tienda online

Función general: Este proyecto consiste en desarrollar un sistema de administración de una tienda online orientado a la web. Los usuarios pueden registrarse en el sistema bajo varios roles, y su interacción con el sistema varían dependiendo del rol asignado.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar productos, categorías, usuarios y pedidos, y supervisar todas las operaciones del sistema.
- Vendedor: Puede añadir, editar y eliminar productos, y gestionar los pedidos.
- Cliente: Puede buscar productos, añadir productos al carrito, realizar pedidos y revisar el estado de los pedidos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Vendedor'), ('Cliente');

CREATE TABLE usuarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contrasena VARCHAR(100) NOT NULL,
  id_rol INT NOT NULL,
  FOREIGN KEY (id_rol) REFERENCES roles(id)
);

CREATE TABLE productos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  descripcion VARCHAR(255) NOT NULL,
  categoria VARCHAR(20) NOT NULL,
  precio DECIMAL(10, 2) NOT NULL,
  stock INT NOT NULL
);

CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_usuario INT NOT NULL,
  id_producto INT NOT NULL,
  cantidad INT NOT NULL,
  fecha_pedido DATE NOT NULL,
  fecha_envio DATE,
  fecha_entrega DATE
);
```

Ejercicio 3

Proyecto: Sistema de gestión de una escuela

Función general: Este proyecto consiste en desarrollar un sistema de gestión de una escuela orientado a la web. Los usuarios pueden registrarse en el sistema bajo varios roles, y su interacción con el sistema varían dependiendo del rol asignado.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar estudiantes, profesores, asignaturas, cursos y calificaciones, y supervisar todas las operaciones del sistema.
- Profesor: Puede añadir, editar y eliminar notas de los estudiantes, y gestionar los datos de los estudiantes.
- Estudiante: Puede ver sus notas, consultar el calendario escolar y realizar solicitudes de matrícula.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Profesor'), ('Estudiante');

CREATE TABLE usuarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  id_rol INT NOT NULL,
  FOREIGN KEY (id_rol) REFERENCES roles(id)
);

CREATE TABLE estudiantes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  id_rol INT NOT NULL,
  FOREIGN KEY (id_rol) REFERENCES roles(id)
);

CREATE TABLE profesores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  id_rol INT NOT NULL,
  FOREIGN KEY (id_rol) REFERENCES roles(id)
);

CREATE TABLE asignaturas (
```

```
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL
);

CREATE TABLE cursos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_asignatura INT NOT NULL,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  fecha_inicio DATE NOT NULL,
  fecha_fin DATE NOT NULL
);

CREATE TABLE calificaciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_estudiante INT NOT NULL,
  id_curso INT NOT NULL,
  calificación FLOAT NOT NULL
);
```

Ejercicio 4

Proyecto: Sistema de gestión de una clínica

Función general: Este proyecto consiste en desarrollar un sistema de gestión de una clínica orientado a la web. Los usuarios pueden registrarse en el sistema bajo varios roles, y su interacción con el sistema varían dependiendo del rol asignado.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar pacientes, médicos, especialistas, citas y recetas, y supervisar todas las operaciones del sistema.
- Médico: Puede añadir, editar y eliminar citas, recetas y notas de los pacientes.
- Paciente: Puede programar citas, consultar su historial médico y ver sus recetas.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Médico'), ('Paciente');

CREATE TABLE usuarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  id_rol INT NOT NULL
);

CREATE TABLE pacientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  id_rol INT NOT NULL
);

CREATE TABLE médicos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  id_rol INT NOT NULL
);

CREATE TABLE citas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_paciente INT NOT NULL,
  id_médico INT NOT NULL,
  fecha_cita DATE NOT NULL,
  hora_cita TIME NOT NULL,
  motivo VARCHAR(255) NOT NULL
```

```
);

CREATE TABLE recetas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_cita INT NOT NULL,
  medicamento VARCHAR(50) NOT NULL,
  dosis VARCHAR(255) NOT NULL,
  frecuencia VARCHAR(255) NOT NULL
);
```

Ejercicio 5

Proyecto: Sistema de gestión de una empresa de transporte

Función general: Este proyecto consiste en desarrollar un sistema de gestión de una empresa de transporte orientado a la web. El sistema debe permitir a la empresa gestionar los pedidos de transporte, los vehículos y los conductores.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar clientes, pedidos de transporte, vehículos y conductores.
- Gestor de pedidos: Puede gestionar los pedidos de transporte, como asignar vehículos y conductores.
- Conductor: Puede aceptar o rechazar pedidos de transporte.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Gestor de pedidos'),
('Conductor');

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  dirección VARCHAR(255) NOT NULL,
  teléfono VARCHAR(20) NOT NULL
);

CREATE TABLE pedidos_transporte (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_cliente INT NOT NULL,
  fecha_inicio DATE NOT NULL,
  fecha_fin DATE NOT NULL,
  origen VARCHAR(255) NOT NULL,
  destino VARCHAR(255) NOT NULL,
  peso FLOAT NOT NULL,
  volumen FLOAT NOT NULL
);

CREATE TABLE vehículos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  tipo VARCHAR(20) NOT NULL,
  capacidad_peso FLOAT NOT NULL,
  capacidad_volumen FLOAT NOT NULL
);

CREATE TABLE conductores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  teléfono VARCHAR(20) NOT NULL,
```

```
  licencia VARCHAR(20) NOT NULL
);

CREATE TABLE asignaciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_pedido_transporte INT NOT NULL,
  id_vehículo INT NOT NULL,
  id_conductor INT NOT NULL
);
```

Ejercicio 6

Proyecto: Sistema de gestión para un restaurante

Función general: Este proyecto consiste en desarrollar un sistema de gestión para un restaurante orientado a la web. El sistema debe permitir al restaurante gestionar los pedidos, la carta, los empleados y los clientes.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar platos, empleados, clientes y pedidos.
- Mesero: Puede tomar pedidos, gestionar las mesas y cobrar a los clientes.
- Cocinero: Puede preparar los pedidos y gestionar la cocina.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Mesero'), ('Cocinero');

CREATE TABLE empleados (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  id_rol INT NOT NULL
);

CREATE TABLE platos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL,
  categoría VARCHAR(20) NOT NULL
);

CREATE TABLE mesas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  capacidad INT NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_mesa INT NOT NULL,
  fecha_hora DATETIME NOT NULL,
  total FLOAT NOT NULL
);

CREATE TABLE pedidos_platos (
  id INT AUTO_INCREMENT PRIMARY KEY,
```

```
  id_pedido INT NOT NULL,
  id_plato INT NOT NULL,
  cantidad INT NOT NULL
);
```

Ejercicio 7

Proyecto: Sistema de citas para una peluquería

Función general: Este proyecto consiste en desarrollar un sistema de citas para una peluquería orientado a la web. El sistema debe permitir a los clientes reservar citas con los peluqueros de la peluquería.

Roles de usuario:

- Cliente: Puede reservar citas, consultar el estado de las citas y cancelar citas.
- Peluquero: Puede ver las citas de los clientes y confirmar o rechazar citas.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Cliente'), ('Peluquero');

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE peluqueros (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE citas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_cliente INT NOT NULL,
  id_peluquero INT NOT NULL,
  fecha_hora DATETIME NOT NULL,
  servicio VARCHAR(255) NOT NULL
);
```

Ejercicio 8

Proyecto: Plataforma de crowdfunding

Función general: Este proyecto consiste en desarrollar una plataforma de crowdfunding orientada a la web. El sistema debe permitir a los usuarios crear campañas de crowdfunding para recaudar fondos para sus proyectos.

Roles de usuario:

- Creador de campaña: Puede crear campañas de crowdfunding, gestionar las aportaciones y comunicar con los patrocinadores.
- Patrocinador: Puede realizar aportaciones a campañas de crowdfunding.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Creador de campaña'), ('Patrocinador');

CREATE TABLE campañas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(255) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  objetivo FLOAT NOT NULL,
  fecha_inicio DATE NOT NULL,
  fecha_fin DATE NOT NULL
);

CREATE TABLE aportaciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_campaña INT NOT NULL,
  id_patrocinador INT NOT NULL,
  cantidad FLOAT NOT NULL
);

CREATE TABLE patrocinadores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);
```

Ejercicio 9

Proyecto: Sistema de administración de un zoológico

Función general: Este proyecto consiste en desarrollar un sistema de administración de un zoológico orientado a la web. El sistema debe permitir al zoológico gestionar los animales, las instalaciones, los empleados y los visitantes.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar animales, instalaciones, empleados y visitantes.
- Veterinario: Puede gestionar la salud de los animales.
- Cuidador: Puede cuidar de los animales.
- Educador: Puede educar a los visitantes sobre los animales.
- Vigilante: Puede controlar el acceso al zoológico y la seguridad de los visitantes.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Veterinario'), ('Cuidador'),
('Educador'), ('Vigilante');

CREATE TABLE animales (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  especie VARCHAR(50) NOT NULL,
  sexo VARCHAR(20) NOT NULL,
  edad INT NOT NULL,
  estado_salud VARCHAR(20) NOT NULL
);

CREATE TABLE instalaciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  tipo VARCHAR(20) NOT NULL,
  capacidad INT NOT NULL
);

CREATE TABLE empleados (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  id_rol INT NOT NULL
);

CREATE TABLE visitantes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
```

```
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE visitas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  id_visitante INT NOT NULL,
  fecha_hora DATETIME NOT NULL
);
```

Ejercicio 10

Proyecto: Sistema de gestión de un gimnasio

Función general: Este proyecto consiste en desarrollar un sistema de gestión de un gimnasio orientado a la web. El sistema debe permitir al gimnasio gestionar los miembros, las clases, los instructores y las instalaciones.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar miembros, clases, instructores e instalaciones.
- Instructor: Puede gestionar sus clases y sus horarios.
- Miembro: Puede reservar clases, consultar su historial de actividades y acceder a su perfil.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Instructor'), ('Miembro');

CREATE TABLE miembros (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  fecha_nacimiento DATE NOT NULL,
  género VARCHAR(20) NOT NULL
);

CREATE TABLE clases (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  tipo VARCHAR(20) NOT NULL,
  instructor_id INT NOT NULL,
  fecha_hora DATETIME NOT NULL
);

CREATE TABLE instructores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  especialidad VARCHAR(20) NOT NULL
);

CREATE TABLE instalaciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  tipo VARCHAR(20) NOT NULL,
  capacidad INT NOT NULL
```

```
);

CREATE TABLE reservas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  miembro_id INT NOT NULL,
  clase_id INT NOT NULL,
  fecha_hora DATETIME NOT NULL
);
```

Ejercicio 11

Proyecto: Sistema de reserva de billetes de avión

Función general: Este proyecto consiste en desarrollar un sistema de reserva de billetes de avión orientado a la web. El sistema debe permitir a los usuarios reservar billetes de avión, consultar el estado de sus reservas y acceder a su historial de reservas.

Roles de usuario:

- Usuario: Puede reservar billetes de avión, consultar el estado de sus reservas y acceder a su historial de reservas.
- Agente de viajes: Puede realizar reservas de billetes de avión en nombre de los usuarios.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Usuario'), ('Agente de viajes');

CREATE TABLE vuelos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  origen VARCHAR(20) NOT NULL,
  destino VARCHAR(20) NOT NULL,
  fecha_salida DATETIME NOT NULL,
  fecha_llegada DATETIME NOT NULL,
  duración INT NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE reservas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  usuario_id INT NOT NULL,
  vuelo_id INT NOT NULL,
  fecha_reserva DATETIME NOT NULL,
  número_pasajeros INT NOT NULL
);

CREATE TABLE pasajeros (
  id INT AUTO_INCREMENT PRIMARY KEY,
  reserva_id INT NOT NULL,
  nombre VARCHAR(50) NOT NULL,
  apellidos VARCHAR(50) NOT NULL,
  fecha_nacimiento DATE NOT NULL,
  género VARCHAR(20) NOT NULL
);
```

Ejercicio 12

Proyecto: Gestor de pedidos para una floristería online

Función general: Este proyecto consiste en desarrollar un gestor de pedidos para una floristería online. El sistema debe permitir a los clientes realizar pedidos de flores, consultar el estado de sus pedidos y acceder a su historial de pedidos.

Roles de usuario:

- Cliente: Puede realizar pedidos de flores, consultar el estado de sus pedidos y acceder a su historial de pedidos.
- Empleado de la floristería: Puede gestionar los pedidos de los clientes, como confirmar los pedidos, preparar los pedidos y enviar los pedidos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Cliente'), ('Empleado');

CREATE TABLE productos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  fecha_pedido DATETIME NOT NULL,
  fecha_envío DATETIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE lineas_pedido (
  id INT AUTO_INCREMENT PRIMARY KEY,
  pedido_id INT NOT NULL,
  producto_id INT NOT NULL,
  cantidad INT NOT NULL
);
```

Ejercicio 13

Proyecto: Sistema de gestión de una inmobiliaria

Función general: Este proyecto consiste en desarrollar un sistema de gestión de una inmobiliaria orientado a la web. El sistema debe permitir a la inmobiliaria gestionar los inmuebles, los clientes y las operaciones inmobiliarias.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar inmuebles, clientes y operaciones inmobiliarias.
- Agente inmobiliario: Puede gestionar los inmuebles y las operaciones inmobiliarias.
- Cliente: Puede buscar inmuebles, ponerse en contacto con agentes inmobiliarios y gestionar sus operaciones inmobiliarias.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Agente inmobiliario'),
('Cliente');

CREATE TABLE inmuebles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  tipo VARCHAR(20) NOT NULL,
  ubicación VARCHAR(20) NOT NULL,
  superficie INT NOT NULL,
  precio FLOAT NOT NULL,
  descripción VARCHAR(255) NOT NULL
);

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE operaciones_inmobiliarias (
  id INT AUTO_INCREMENT PRIMARY KEY,
  tipo VARCHAR(20) NOT NULL,
  inmueble_id INT NOT NULL,
  cliente_id INT NOT NULL,
  fecha_inicio DATETIME NOT NULL,
  fecha_fin DATETIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);
```

Ejercicio 14

Proyecto: Sistema de gestión de una granja

Función general: Este proyecto consiste en desarrollar un sistema de gestión de una granja orientado a la web. El sistema debe permitir a la granja gestionar los animales, las instalaciones, los empleados y los cultivos.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar animales, instalaciones, empleados y cultivos.
- Encargado de la granja: Puede gestionar los animales, las instalaciones y los cultivos.
- Empleados de la granja: Pueden realizar tareas específicas, como alimentar a los animales o cosechar los cultivos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Encargado de la granja'),
('Empleado');

CREATE TABLE animales (
  id INT AUTO_INCREMENT PRIMARY KEY,
  especie VARCHAR(20) NOT NULL,
  raza VARCHAR(20) NOT NULL,
  sexo VARCHAR(20) NOT NULL,
  edad INT NOT NULL,
  estado_salud VARCHAR(20) NOT NULL
);

CREATE TABLE instalaciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  tipo VARCHAR(20) NOT NULL,
  capacidad INT NOT NULL
);

CREATE TABLE empleados (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  fecha_nacimiento DATE NOT NULL,
  género VARCHAR(20) NOT NULL
);

CREATE TABLE cultivos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  tipo VARCHAR(20) NOT NULL,
  variedad VARCHAR(20) NOT NULL,
```

```
  fecha_siembra DATE NOT NULL,
  fecha_cosecha DATE NOT NULL
);
```

Ejercicio 15

Proyecto: Sistema de administración de un museo

Función general: Este proyecto consiste en desarrollar un sistema de administración de un museo orientado a la web. El sistema debe permitir al museo gestionar las colecciones, las exposiciones, los visitantes y el personal.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar colecciones, exposiciones, visitantes y personal.
- Curador: Puede gestionar las colecciones y las exposiciones.
- Visitante: Puede consultar las colecciones y las exposiciones.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Curador'), ('Visitante');

CREATE TABLE colecciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL
);

CREATE TABLE obras (
  id INT AUTO_INCREMENT PRIMARY KEY,
  colección_id INT NOT NULL,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  autor VARCHAR(50) NOT NULL,
  fecha_creación DATE NOT NULL
);

CREATE TABLE exposiciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  fecha_inicio DATE NOT NULL,
  fecha_fin DATE NOT NULL
);

CREATE TABLE visitantes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE personal (
```

```
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  fecha_nacimiento DATE NOT NULL,
  género VARCHAR(20) NOT NULL
);
```

Ejercicio 16

Proyecto: Sistema de gestión para una agencia de viajes

Función general: Este proyecto consiste en desarrollar un sistema de gestión para una agencia de viajes orientado a la web. El sistema debe permitir a la agencia de viajes gestionar los clientes, los viajes, las reservas y los pagos.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar clientes, viajes, reservas y pagos.
- Agente de viajes: Puede gestionar los clientes, los viajes y las reservas.
- Cliente: Puede consultar los viajes, realizar reservas y realizar pagos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Agente de viajes'), ('Cliente');

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE viajes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  destino VARCHAR(50) NOT NULL,
  duración INT NOT NULL,
  precio FLOAT NOT NULL,
  descripción VARCHAR(255) NOT NULL
);

CREATE TABLE reservas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  viaje_id INT NOT NULL,
  fecha_inicio DATE NOT NULL,
  fecha_fin DATE NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE pagos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  reserva_id INT NOT NULL,
  fecha DATETIME NOT NULL,
  importe FLOAT NOT NULL
);
```

Ejercicio 17

Proyecto: Sistema de administración para una empresa de limpieza

Función general: Este proyecto consiste en desarrollar un sistema de administración para una empresa de limpieza orientado a la web. El sistema debe permitir a la empresa de limpieza gestionar los clientes, los empleados, los servicios, los pedidos y los pagos.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar clientes, empleados, servicios, pedidos y pagos.
- Ejecutivo de ventas: Puede gestionar los clientes, los servicios y los pedidos.
- Empleado de limpieza: Puede consultar los pedidos y realizar el trabajo de limpieza.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Ejecutivo de ventas'), ('Empleado
de limpieza');

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE empleados (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  fecha_nacimiento DATE NOT NULL,
  género VARCHAR(20) NOT NULL
);

CREATE TABLE servicios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  empleado_id INT NOT NULL,
  servicio_id INT NOT NULL,
```

```
  fecha DATE NOT NULL,
  hora DATETIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE pagos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  pedido_id INT NOT NULL,
  fecha DATETIME NOT NULL,
  importe FLOAT NOT NULL
);
```

Ejercicio 18

Proyecto: Plataforma de gestión para una ONG

Función general: Este proyecto consiste en desarrollar una plataforma de gestión para una ONG orientada a la web. La plataforma debe permitir a la ONG gestionar los voluntarios, las donaciones, las campañas y los proyectos.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar voluntarios, donaciones, campañas y proyectos.
- Voluntario: Puede consultar las campañas y proyectos de la ONG.
- Donante: Puede realizar donaciones a la ONG.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Voluntario'), ('Donante');

CREATE TABLE voluntarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL,
  habilidades VARCHAR(255) NOT NULL
);

CREATE TABLE donaciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  donante_id INT NOT NULL,
  importe FLOAT NOT NULL,
  fecha DATE NOT NULL
);

CREATE TABLE campañas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  objetivo FLOAT NOT NULL,
  fecha_inicio DATE NOT NULL,
  fecha_fin DATE NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE proyectos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  campaña_id INT NOT NULL,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
```

```
  fecha_inicio DATE NOT NULL,
  fecha_fin DATE NOT NULL,
  estado VARCHAR(20) NOT NULL
);
```

Ejercicio 19

Proyecto: Sistema de gestión para un salón de belleza

Función general: Este proyecto consiste en desarrollar un sistema de gestión para un salón de belleza orientado a la web. El sistema debe permitir al salón de belleza gestionar los clientes, los servicios, las citas, los empleados y las finanzas.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar clientes, servicios, citas, empleados y finanzas.
- Empleado: Puede gestionar las citas y los servicios.
- Cliente: Puede consultar las citas y los servicios.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Empleado'), ('Cliente');

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE servicios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE citas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  empleado_id INT NOT NULL,
  servicio_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE empleados (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  fecha_nacimiento DATE NOT NULL,
```

```
  género VARCHAR(20) NOT NULL
);

CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  ingresos FLOAT NOT NULL,
  gastos FLOAT NOT NULL,
  balance FLOAT NOT NULL
);
```

Ejercicio 20

Proyecto: Sistema de administración de un spa

Función general: Este proyecto consiste en desarrollar un sistema de administración para un spa orientado a la web. El sistema debe permitir al spa gestionar los clientes, los servicios, las reservas, los empleados y las finanzas.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar clientes, servicios, reservas, empleados y finanzas.
- Técnico: Puede gestionar las reservas y los servicios.
- Cliente: Puede consultar las reservas y los servicios.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Técnico'), ('Cliente');

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE servicios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE reservas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  técnico_id INT NOT NULL,
  servicio_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE empleados (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  fecha_nacimiento DATE NOT NULL,
```

```
  género VARCHAR(20) NOT NULL
);

CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  ingresos FLOAT NOT NULL,
  gastos FLOAT NOT NULL,
  balance FLOAT NOT NULL
);
```

Ejercicio 21

Proyecto: Sistema de citas para una clínica dental

Función general: Este proyecto consiste en desarrollar un sistema de citas para una clínica dental orientado a la web. El sistema debe permitir a la clínica dental gestionar las citas, los pacientes, los tratamientos y los empleados.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar pacientes, tratamientos, citas y empleados.
- Recepcionista: Puede gestionar las citas y los pacientes.
- Dentista: Puede realizar tratamientos.
- Paciente: Puede consultar las citas y los tratamientos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Recepcionista'), ('Dentista'),
('Paciente');

CREATE TABLE pacientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL,
  fecha_nacimiento DATE NOT NULL
);

CREATE TABLE tratamientos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE citas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  paciente_id INT NOT NULL,
  dentista_id INT NOT NULL,
  tratamiento_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE empleados (
  id INT AUTO_INCREMENT PRIMARY KEY,
```

```
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  fecha_nacimiento DATE NOT NULL,
  género VARCHAR(20) NOT NULL,
  rol_id INT NOT NULL
);

ALTER TABLE empleados
ADD CONSTRAINT fk_empleados_roles
FOREIGN KEY (rol_id) REFERENCES roles (id);
```

Ejercicio 22

Proyecto: Sistema de reservas para un campo de golf

Función general: Este proyecto consiste en desarrollar un sistema de reservas para un campo de golf orientado a la web. El sistema debe permitir a los jugadores reservar tee times, consultar los horarios de juego y las tarifas.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar jugadores, tee times, horarios de juego y tarifas.
- Jugador: Puede reservar tee times y consultar los horarios de juego y las tarifas.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Jugador');

CREATE TABLE jugadores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE tee_times (
  id INT AUTO_INCREMENT PRIMARY KEY,
  jugador_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  greenfee FLOAT NOT NULL
);

CREATE TABLE horarios_juego (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  hora_inicio TIME NOT NULL,
  hora_fin TIME NOT NULL
);

CREATE TABLE tarifas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  greenfee FLOAT NOT NULL
);
```

Ejercicio 23

Proyecto: Sistema de administración para una boda

Función general: Este proyecto consiste en desarrollar un sistema de administración para una boda orientado a la web. El sistema debe permitir a los novios organizar su boda, gestionar los invitados, el presupuesto y los proveedores.

Roles de usuario:

- Novio: Puede añadir, editar y eliminar invitados, proveedores, gastos y tareas.
- Novia: Puede añadir, editar y eliminar invitados, proveedores, gastos y tareas.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Novio'), ('Novia');

CREATE TABLE invitados (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  teléfono VARCHAR(10) NOT NULL,
  relación VARCHAR(20) NOT NULL
);

CREATE TABLE proveedores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  teléfono VARCHAR(10) NOT NULL,
  tipo VARCHAR(20) NOT NULL
);

CREATE TABLE gastos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  proveedor_id INT NOT NULL,
  fecha DATE NOT NULL,
  importe FLOAT NOT NULL
);

CREATE TABLE tareas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  título VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  fecha_inicio DATE NOT NULL,
  fecha_fin DATE NOT NULL,
  estado VARCHAR(20) NOT NULL
);
```

Ejercicio 24

Proyecto: Sistema de gestión para un salón de eventos

Función general: Este proyecto consiste en desarrollar un sistema de gestión para un salón de eventos orientado a la web. El sistema debe permitir al salón de eventos gestionar los eventos, los clientes, los proveedores y las finanzas.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar eventos, clientes, proveedores y finanzas.
- Empleado: Puede gestionar los eventos, los clientes y los proveedores.
- Cliente: Puede consultar los eventos disponibles.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Empleado'), ('Cliente');

CREATE TABLE eventos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  capacidad INT NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE proveedores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  teléfono VARCHAR(10) NOT NULL,
  tipo VARCHAR(20) NOT NULL
);

CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  ingresos FLOAT NOT NULL,
  gastos FLOAT NOT NULL,
  balance FLOAT NOT NULL
```

```
);
```

Ejercicio 25

Proyecto: Sistema de citas para un taller de reparación de automóviles

Función general: Este proyecto consiste en desarrollar un sistema de citas para un taller de reparación de automóviles orientado a la web. El sistema debe permitir al taller de reparación de automóviles gestionar las citas, los clientes, los vehículos y las reparaciones.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar citas, clientes, vehículos y reparaciones.
- Mecánico: Puede gestionar las citas y las reparaciones.
- Cliente: Puede consultar las citas y los vehículos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Mecánico'), ('Cliente');

CREATE TABLE citas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  vehículo_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE vehículos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  marca VARCHAR(50) NOT NULL,
  modelo VARCHAR(50) NOT NULL,
  año INT NOT NULL,
  patente VARCHAR(10) NOT NULL
);

CREATE TABLE reparaciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cita_id INT NOT NULL,
  descripción VARCHAR(255) NOT NULL,
```

```
  precio FLOAT NOT NULL
);
```

Ejercicio 26

Proyecto: Sistema de administración de un club deportivo

Función general: Este proyecto consiste en desarrollar un sistema de administración de un club deportivo orientado a la web. El sistema debe permitir al club deportivo gestionar los socios, los equipos, las competiciones y las instalaciones.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar socios, equipos, competiciones e instalaciones.
- Entrenador: Puede gestionar los equipos y las competiciones.
- Jugador: Puede consultar su información personal y la de su equipo.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Entrenador'), ('Jugador');

CREATE TABLE socios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE equipos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  deporte VARCHAR(20) NOT NULL,
  categoría VARCHAR(20) NOT NULL
);

CREATE TABLE competiciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  deporte VARCHAR(20) NOT NULL,
  categoría VARCHAR(20) NOT NULL,
  fecha_inicio DATE NOT NULL,
  fecha_fin DATE NOT NULL
);

CREATE TABLE instalaciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  tipo VARCHAR(20) NOT NULL,
  capacidad INT NOT NULL
);
```

Ejercicio 27

Proyecto: Sistema de reservas para un estudio de fotografía

Función general: Este proyecto consiste en desarrollar un sistema de reservas para un estudio de fotografía orientado a la web. El sistema debe permitir a los clientes reservar sesiones de fotografía en el estudio.

Roles de usuario:

- Cliente: Puede reservar sesiones de fotografía.
- Empleado: Puede gestionar las reservas y los servicios fotográficos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Cliente'), ('Empleado');

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE servicios_fotográficos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE reservas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  servicio_fotográfico_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);
```

Ejercicio 28

Proyecto: Sistema de gestión para una ferretería online

Función general: Este proyecto consiste en desarrollar un sistema de gestión para una ferretería online orientado a la web. El sistema debe permitir a la ferretería gestionar los productos, los pedidos, los clientes y las finanzas.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar productos, pedidos, clientes y finanzas.
- Empleado: Puede gestionar los pedidos y los clientes.
- Cliente: Puede realizar pedidos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Empleado'), ('Cliente');

CREATE TABLE productos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL,
  stock INT NOT NULL
);

CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  fecha DATE NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE pedidos_productos (
  pedido_id INT NOT NULL,
  producto_id INT NOT NULL,
  cantidad INT NOT NULL
);

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
```

```
  fecha DATE NOT NULL,
  ingresos FLOAT NOT NULL,
  gastos FLOAT NOT NULL,
  balance FLOAT NOT NULL
);
```

Ejercicio 29

Proyecto: Sistema de administración de una guardería

Función general: Este proyecto consiste en desarrollar un sistema de administración de una guardería orientado a la web. El sistema debe permitir a la guardería gestionar a los niños, los padres, los empleados, los horarios y las actividades.

Roles de usuario:

- Director: Puede añadir, editar y eliminar niños, padres, empleados, horarios y actividades.
- Educador: Puede gestionar a los niños y las actividades.
- Padre: Puede consultar la información de su hijo y los horarios.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Director'), ('Educador'), ('Padre');

CREATE TABLE niños (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  fecha_nacimiento DATE NOT NULL,
  alergias VARCHAR(255) NOT NULL,
  medicación VARCHAR(255) NOT NULL
);

CREATE TABLE padres (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE empleados (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL,
  teléfono VARCHAR(10) NOT NULL
);

CREATE TABLE horarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  dia VARCHAR(20) NOT NULL,
  hora_inicio TIME NOT NULL,
  hora_fin TIME NOT NULL
);
```

```
CREATE TABLE actividades (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  horario_id INT NOT NULL
);

CREATE TABLE asistencias (
  niño_id INT NOT NULL,
  horario_id INT NOT NULL,
  fecha DATE NOT NULL,
  asistencia VARCHAR(20) NOT NULL
);
```

Ejercicio 30

Proyecto: Plataforma de gestión para un sitio de noticias

Función general: Este proyecto consiste en desarrollar una plataforma de gestión para un sitio de noticias orientado a la web. La plataforma debe permitir a los editores de noticias gestionar el contenido, los usuarios y las configuraciones del sitio.

Roles de usuario:

- Editor: Puede añadir, editar y eliminar noticias, categorías, usuarios y configuraciones.
- Usuario: Puede leer noticias y comentarlas.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Editor'), ('Usuario');

CREATE TABLE noticias (
  id INT AUTO_INCREMENT PRIMARY KEY,
  titulo VARCHAR(255) NOT NULL,
  contenido TEXT NOT NULL,
  fecha_publicacion DATE NOT NULL,
  categoria_id INT NOT NULL
);

CREATE TABLE categorias (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(255) NOT NULL
);

CREATE TABLE usuarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE configuraciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(255) NOT NULL,
  valor VARCHAR(255) NOT NULL
);
```

Ejercicio 31

Proyecto: Sistema de reservas para una sala de cine

Función general: Este proyecto consiste en desarrollar un sistema de reservas para una sala de cine orientado a la web. El sistema debe permitir a los clientes reservar entradas para las películas que se proyectan en la sala de cine.

Roles de usuario:

- Cliente: Puede reservar entradas para las películas.
- Administrador: Puede gestionar las películas, las sesiones y las reservas.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Cliente'), ('Administrador');

CREATE TABLE películas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  título VARCHAR(255) NOT NULL,
  sinopsis VARCHAR(255) NOT NULL,
  género VARCHAR(20) NOT NULL,
  duración INT NOT NULL,
  clasificación_edad VARCHAR(20) NOT NULL
);

CREATE TABLE sesiones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  película_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  sala_id INT NOT NULL
);

CREATE TABLE salas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  capacidad INT NOT NULL
);

CREATE TABLE reservas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  sesión_id INT NOT NULL,
  asientos VARCHAR(255) NOT NULL
);
```

Ejercicio 32

Proyecto: Sistema de gestión de una lavandería

Función general: Este proyecto consiste en desarrollar un sistema de gestión para una lavandería orientado a la web. El sistema debe permitir a la lavandería gestionar los clientes, las máquinas, las tarifas y los pedidos.

Roles de usuario:

- Cliente: Puede realizar pedidos de lavado y planchado.
- Administrador: Puede gestionar los clientes, las máquinas, las tarifas y los pedidos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Cliente'), ('Administrador');

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE máquinas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  tipo VARCHAR(20) NOT NULL,
  capacidad INT NOT NULL
);

CREATE TABLE tarifas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  tipo VARCHAR(20) NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  máquina_id INT NOT NULL,
  tarifa_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL
);
```

Ejercicio 33

Proyecto: Sistema de gestión de una panadería

Función general: Este proyecto consiste en desarrollar un sistema de gestión para una panadería orientado a la web. El sistema debe permitir a la panadería gestionar los productos, los pedidos, los clientes y las finanzas.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar productos, pedidos, clientes y finanzas.
- Empleado: Puede gestionar los pedidos y los clientes.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Empleado');

CREATE TABLE productos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL,
  stock INT NOT NULL
);

CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  producto_id INT NOT NULL,
  cantidad INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL
);

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  ingresos FLOAT NOT NULL,
  gastos FLOAT NOT NULL,
  balance FLOAT NOT NULL
);
```

Ejercicio 34

Proyecto: Sistema de gestión de una pizzería

Función general: Este proyecto consiste en desarrollar un sistema de gestión para una pizzería orientado a la web. El sistema debe permitir a la pizzería gestionar los productos, los pedidos, los clientes y las finanzas.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar productos, pedidos, clientes y finanzas.
- Empleado: Puede gestionar los pedidos y los clientes.
- Cliente: Puede realizar pedidos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Empleado'), ('Cliente');

CREATE TABLE productos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL,
  stock INT NOT NULL
);

CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  producto_id INT NOT NULL,
  cantidad INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  ingresos FLOAT NOT NULL,
  gastos FLOAT NOT NULL,
  balance FLOAT NOT NULL
);
```

Ejercicio 35

Proyecto: Sistema de administración para una cervecera artesanal

Función general: Este proyecto consiste en desarrollar un sistema de administración para una cervecera artesanal orientado a la web. El sistema debe permitir a la cervecera gestionar los productos, los pedidos, los clientes y las finanzas.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar productos, pedidos, clientes y finanzas.
- Empleado: Puede gestionar los pedidos y los clientes.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Empleado');

CREATE TABLE productos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  tipo VARCHAR(20) NOT NULL,
  precio FLOAT NOT NULL,
  stock INT NOT NULL
);

CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  producto_id INT NOT NULL,
  cantidad INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  ingresos FLOAT NOT NULL,
  gastos FLOAT NOT NULL,
  balance FLOAT NOT NULL
);
```

Ejercicio 36

Proyecto: Plataforma de gestión para un podcast

Función general: Este proyecto consiste en desarrollar una plataforma de gestión para un podcast orientado a la web. La plataforma debe permitir al podcaster gestionar los episodios, las notas, los invitados, los suscriptores y las estadísticas.

Roles de usuario:

- Podcaster: Puede gestionar los episodios, las notas, los invitados, los suscriptores y las estadísticas.
- Oyente: Puede escuchar los episodios y suscribirse al podcast.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Podcaster'), ('Oyente');

CREATE TABLE episodios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  título VARCHAR(255) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  fecha_publicación DATE NOT NULL,
  duración INT NOT NULL,
  archivo_audio BLOB NOT NULL
);

CREATE TABLE notas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  episodio_id INT NOT NULL,
  título VARCHAR(255) NOT NULL,
  contenido VARCHAR(255) NOT NULL
);

CREATE TABLE invitados (
  id INT AUTO_INCREMENT PRIMARY KEY,
  episodio_id INT NOT NULL,
  nombre VARCHAR(255) NOT NULL,
  biografía VARCHAR(255) NOT NULL
);

CREATE TABLE suscriptores (
  id INT AUTO_INCREMENT PRIMARY KEY,
  correo VARCHAR(100) UNIQUE NOT NULL
);

CREATE TABLE estadísticas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  episodio_id INT NOT NULL,
  reproducciones INT NOT NULL,
  descargas INT NOT NULL
```

```
);
```

Ejercicio 37

Proyecto: Sistema de gestión de una heladería

Función general: Este proyecto consiste en desarrollar un sistema de gestión para una heladería orientado a la web. El sistema debe permitir a la heladería gestionar los productos, los pedidos, los clientes y las finanzas.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar productos, pedidos, clientes y finanzas.
- Empleado: Puede gestionar los pedidos y los clientes.
- Cliente: Puede realizar pedidos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Empleado'), ('Cliente');

CREATE TABLE productos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL,
  stock INT NOT NULL
);

CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  producto_id INT NOT NULL,
  cantidad INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  ingresos FLOAT NOT NULL,
  gastos FLOAT NOT NULL,
  balance FLOAT NOT NULL
);
```

Ejercicio 38

Proyecto: Sistema de administración para un granjero de mariscos

Función general: Este proyecto consiste en desarrollar un sistema de administración para un granjero de mariscos orientado a la web. El sistema debe permitir al granjero gestionar las granjas, los cultivos, los suministros, los empleados y las finanzas.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar granjas, cultivos, suministros, empleados y finanzas.
- Empleado: Puede gestionar las granjas, los cultivos y los suministros.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Empleado');

CREATE TABLE granjas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  ubicación VARCHAR(255) NOT NULL,
  tipo VARCHAR(20) NOT NULL
);

CREATE TABLE cultivos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  granja_id INT NOT NULL,
  tipo VARCHAR(20) NOT NULL,
  cantidad INT NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE suministros (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  cantidad INT NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE empleados (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
```

```
  ingresos FLOAT NOT NULL,
  gastos FLOAT NOT NULL,
  balance FLOAT NOT NULL
);
```

Ejercicio 39

Proyecto: Sistema de reservas para una peluquería canina

Función general: Este proyecto consiste en desarrollar un sistema de reservas para una peluquería canina orientado a la web. El sistema debe permitir a los clientes reservar servicios para sus perros.

Roles de usuario:

- Cliente: Puede realizar reservas y consultar el estado de sus reservas.
- Administrador: Puede gestionar las reservas, los servicios y los empleados.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Cliente'), ('Administrador');

CREATE TABLE servicios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE reservas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  servicio_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE empleados (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);
```

Ejercicio 40

Proyecto: Sistema de gestión para una joyería

Función general: Este proyecto consiste en desarrollar un sistema de gestión para una joyería orientado a la web. El sistema debe permitir a la joyería gestionar los productos, los clientes, los pedidos y las finanzas.

Roles de usuario:

- Administrador: Puede añadir, editar y eliminar productos, clientes, pedidos y finanzas.
- Empleado: Puede gestionar los productos, los clientes y los pedidos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Empleado');

CREATE TABLE productos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL,
  stock INT NOT NULL
);

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  contraseña VARCHAR(100) NOT NULL
);

CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  producto_id INT NOT NULL,
  cantidad INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  estado VARCHAR(20) NOT NULL
);

CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  ingresos FLOAT NOT NULL,
  gastos FLOAT NOT NULL,
  balance FLOAT NOT NULL
);
```

Ejercicio 41

Proyecto: Sistema de gestión para un taller de bicicletas

Función general: Este proyecto consiste en desarrollar un sistema de gestión para un taller de bicicletas orientado a la web. El sistema debe permitir al taller gestionar los clientes, las reparaciones, el inventario y las finanzas.

Roles de usuario:

- Administrador: Puede gestionar todos los aspectos del taller.
- Mecánico: Puede gestionar las reparaciones.
- Recepcionista: Puede gestionar los clientes y el inventario.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Mecánico'), ('Recepcionista');

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  teléfono VARCHAR(10) NOT NULL,
  dirección VARCHAR(255) NOT NULL
);

CREATE TABLE reparaciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  fecha_entrada DATE NOT NULL,
  fecha_salida DATE NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE inventario (
  id INT AUTO_INCREMENT PRIMARY KEY,
  producto VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  cantidad INT NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  ingresos FLOAT NOT NULL,
  gastos FLOAT NOT NULL,
  balance FLOAT NOT NULL
);
```

Ejercicio 42

Proyecto: Sistema de administración de un vivero de plantas

Función general: Este proyecto consiste en desarrollar un sistema de administración para un vivero de plantas orientado a la web. El sistema debe permitir al vivero gestionar los productos, los clientes, las ventas y las finanzas.

Roles de usuario:

- Administrador: Puede gestionar todos los aspectos del vivero.
- Vendedor: Puede gestionar las ventas y los clientes.
- Encargado: Puede gestionar los productos y el inventario.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Vendedor'), ('Encargado');

CREATE TABLE productos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL,
  stock INT NOT NULL
);

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  teléfono VARCHAR(10) NOT NULL,
  dirección VARCHAR(255) NOT NULL
);

CREATE TABLE ventas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  producto_id INT NOT NULL,
  cantidad INT NOT NULL,
  fecha DATE NOT NULL,
  total FLOAT NOT NULL
);

CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  ingresos FLOAT NOT NULL,
  gastos FLOAT NOT NULL,
  balance FLOAT NOT NULL
);
```

Ejercicio 43

Proyecto: Sistema de reservas para un parque de atracciones

Función general: Este proyecto consiste en desarrollar un sistema de reservas para un parque de atracciones orientado a la web. El sistema debe permitir a los clientes reservar entradas, atracciones y servicios.

Roles de usuario:

- Cliente: Puede reservar entradas, atracciones y servicios.
- Administrador: Puede gestionar las reservas, las atracciones y los servicios.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Cliente'), ('Administrador');

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  teléfono VARCHAR(10) NOT NULL,
  dirección VARCHAR(255) NOT NULL
);

CREATE TABLE atracciones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE servicios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE reservas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  atracciones VARCHAR(255) NOT NULL,
  servicios VARCHAR(255) NOT NULL
);
```

Ejercicio 44

Proyecto: Sistema de gestión para una empresa de reciclaje

Función general: Este proyecto consiste en desarrollar un sistema de gestión para una empresa de reciclaje orientado a la web. El sistema debe permitir a la empresa gestionar los clientes, los materiales, las rutas y los residuos.

Roles de usuario:

- Administrador: Puede gestionar todos los aspectos de la empresa.
- Operador: Puede gestionar los materiales, las rutas y los residuos.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Operador');

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  teléfono VARCHAR(10) NOT NULL,
  dirección VARCHAR(255) NOT NULL
);

CREATE TABLE materiales (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL
);

CREATE TABLE rutas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  origen VARCHAR(255) NOT NULL,
  destino VARCHAR(255) NOT NULL,
  distancia INT NOT NULL
);

CREATE TABLE residuos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  material_id INT NOT NULL,
  cantidad INT NOT NULL,
  fecha DATE NOT NULL
);
```

Ejercicio 45

Proyecto: Sistema de administración de un acuario

Función general: Este proyecto consiste en desarrollar un sistema de administración para un acuario orientado a la web. El sistema debe permitir al acuario gestionar los peces, los tanques, los visitantes y las finanzas.

Roles de usuario:

- Administrador: Puede gestionar todos los aspectos del acuario.
- Encargado de peces: Puede gestionar los peces y los tanques.
- Encargado de visitantes: Puede gestionar los visitantes y las finanzas.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Encargado de peces'), ('Encargado de visitantes');

CREATE TABLE peces (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  especie VARCHAR(50) NOT NULL,
  tamaño INT NOT NULL,
  edad INT NOT NULL,
  sexo VARCHAR(10) NOT NULL
);

CREATE TABLE tanques (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  capacidad INT NOT NULL,
  temperatura INT NOT NULL,
  pH INT NOT NULL
);

CREATE TABLE visitantes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  teléfono VARCHAR(10) NOT NULL,
  dirección VARCHAR(255) NOT NULL
);

CREATE TABLE entradas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  visitante_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL
);
```

```
CREATE TABLE finanzas (
  id INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE NOT NULL,
  ingresos INT NOT NULL,
  gastos INT NOT NULL,
  balance INT NOT NULL
);
```

Ejercicio 46

Proyecto: Plataforma para un servicio de entregas de comida a domicilio

Función general: Este proyecto consiste en desarrollar una plataforma para un servicio de entregas de comida a domicilio orientada a la web. La plataforma debe permitir a los clientes realizar pedidos de comida a restaurantes locales y a los restaurantes gestionar sus pedidos y entregas.

Roles de usuario:

- Cliente: Puede realizar pedidos de comida a restaurantes locales.
- Restaurante: Puede gestionar sus pedidos y entregas.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Cliente'), ('Restaurante');

CREATE TABLE restaurantes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  dirección VARCHAR(255) NOT NULL,
  teléfono VARCHAR(10) NOT NULL,
  horario VARCHAR(255) NOT NULL
);

CREATE TABLE platos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  restaurante_id INT NOT NULL,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL,
  precio FLOAT NOT NULL
);

CREATE TABLE pedidos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  restaurante_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  platos VARCHAR(255) NOT NULL,
  estado VARCHAR(20) NOT NULL
);
```

Ejercicio 47

Proyecto: Sistema de administración para un estudio de grabación musical

Función general: Este proyecto consiste en desarrollar un sistema de administración para un estudio de grabación musical orientado a la web. El sistema debe permitir al estudio gestionar los clientes, las sesiones de grabación, los músicos y los equipos.

Roles de usuario:

- Administrador: Puede gestionar todos los aspectos del estudio.
- Ingeniero de sonido: Puede gestionar las sesiones de grabación y los músicos.
- Recepcionista: Puede gestionar los clientes y las citas.

Base de datos:

```
SQL
CREATE TABLE roles (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(20) NOT NULL
);

INSERT INTO roles (nombre) VALUES ('Administrador'), ('Ingeniero de sonido'),
('Recepcionista');

CREATE TABLE clientes (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  correo VARCHAR(100) UNIQUE NOT NULL,
  teléfono VARCHAR(10) NOT NULL,
  dirección VARCHAR(255) NOT NULL
);

CREATE TABLE músicos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  instrumento VARCHAR(50) NOT NULL,
  experiencia INT NOT NULL
);

CREATE TABLE sesiones (
  id INT AUTO_INCREMENT PRIMARY KEY,
  cliente_id INT NOT NULL,
  fecha DATE NOT NULL,
  hora TIME NOT NULL,
  duración INT NOT NULL
);

CREATE TABLE equipos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(50) NOT NULL,
  descripción VARCHAR(255) NOT NULL
);
```

- **Bibliografía**

Autor	Título	Año	Idioma	Edición
Robin Nixon	Learning PHP, MySQL, JavaScript, CSS & HTML5	2014	Inglés	3ra
Hernán Beati	PHP Creación de web dinámicas	2011	Español	NN
Hector Flores Fernández y Jorge Hernández Rodríguez	Aplicaciones web con PHP	2020	Español	2021

- **Orientaciones metodológicas generales**

  Son las siguientes:

➔ El punto de partida para realizar estas actividades lo constituye la revisión del material de estudio, actividad lectora, ideas claves y glosario de la unidad. A su vez, el empleo de un editor web como VS-Code o Sublime Text, como también un entorno WAMP, LAMP ó MAMP como Laragon ó XAMPP.

➔ Del material de estudio, los insumos de la unidad, dan una introducción base al respecto para resolver los ejercicios planteados..

➔ Del mismo material de estudio tenemos un apoyo importante en el desarrollo de código en los archivos pdf: 01_Desarrollo de aplicaciones web con PHP y MySQL y 02_Aplicaciones Web con PHP.

➔ La entrega de la clase práctica, consiste en escribir dos respuestas en el gestor de aprendizaje eva.ube.edu.ec de la asignatura:

  ◆ Considere que el ejercicio es de trabajarse en el grupo de la asignatura, para lo cual el grupo crea un espacio en coda.io y envía el enlace respectivo al docente de la asignatura, para participar en el grupo. Con Version History se podrá visualizar la participación de los integrantes de grupo.

- La dirección URL GitHub del repositorio que cree en una cuenta GitHub como grupo, de la tarea, cuyo nombre sea pwapractica5, de modo que la URL que se genera aplica el siguiente formato: github.com/**grupo99**/pwapractica6 (donde 99 será 01 o 02, o 03, … o 10). Dentro del repositorio estará la aplicación del sistema.
- La dirección URL del sitio web que haya creado, sea en un sitio de hospedaje gratis como: https://es.000webhost.com/, https://www.freehosting.com/, https://www.infinityfree.com/, u otro de su elección, como también lo puede hacer de algún hospedaje privado que disponga. Este sitio web tendrá la lógica indicada en el repositorio GitHub de la tarea, en la forma que se ha indicado previamente.

➔ Como instrumento de evaluación es necesario tomar en cuenta la rúbrica de evaluación de la tarea.

- **Rúbrica, lista de cotejo u otro instrumento para evaluar la tarea.**

Aspecto	Insuficiente (0-8 puntos)	Satisfactorio (9-16 puntos)	Excelente (20 puntos)
Comprensión de PHP	El estudiante muestra dificultad para comprender los conceptos básicos de PHP.	El estudiante tiene una comprensión básica de la sintaxis y estructuras de PHP, pero puede cometer errores.	El estudiante demuestra una comprensión clara de PHP, utiliza funciones, include, require y variables de manera efectiva y desarrolla aplicaciones PHP funcionales.
Comprensión de CSS	El estudiante muestra dificultad para comprender los conceptos básicos de CSS.	El estudiante tiene una comprensión básica de las reglas CSS, pero puede cometer errores en la aplicación.	El estudiante demuestra una comprensión sólida de las reglas CSS, aplicando estilos de manera efectiva y consistente.
Comprensión de JavaScript	El estudiante muestra dificultad para comprender los	El estudiante tiene una comprensión básica de las estructuras y sintaxis	El estudiante demuestra una comprensión sólida de JavaScript, utilizando lógica efectiva para crear interacciones y

	conceptos básicos de JavaScript.	de JavaScript, pero puede cometer errores.	funcionalidades dinámicas en la página.
Desarrollo PHP	El estudiante no puede desarrollar aplicaciones PHP funcionales.	El estudiante puede crear aplicaciones PHP simples, pero con limitaciones en la funcionalidad o la organización del código.	El estudiante es capaz de desarrollar aplicaciones PHP completas, funcionales y bien estructuradas
Estilos y Diseño (CSS)	El estudiante no aplica estilos CSS a la página web o el diseño es poco atractivo.	El estudiante aplica algunos estilos CSS, pero con falta de cohesión en el diseño.	El estudiante crea una interfaz de usuario atractiva y coherente utilizando CSS.

La rúbrica presenta una evaluación sobre 100 puntos.