

Examen Final

1.- Crea la BDD prueba practica y en ella las siguientes tablas, utilizando los tipos y tamaños de datos más adecuados a cada campo y sus correspondientes relaciones:

- Comunidades
 - id_comunidad
 - Nombre
 - Dirección
 - Población
 - id_administrador
- Vecinos
 - id_vecino
 - Nombre
 - Escalera
 - Piso
 - Puerta
 - id comunidad
- Administradores
 - id_administrador
 - Nombre

```
MySQL [(none)]> CREATE DATABASE pruebapractica;
Query OK, 1 row affected (0.005 sec)

MySQL [(none)]> USE pruebapractica;
Database changed
MySQL [pruebapractica]> CREATE TABLE Administradores (
  ->     id_administrador INT AUTO_INCREMENT PRIMARY KEY,
  ->     Nombre VARCHAR(100) NOT NULL
  -> );
Query OK, 0 rows affected (0.021 sec)

MySQL [pruebapractica]> CREATE TABLE Comunidades (
  ->     id_comunidad INT AUTO_INCREMENT PRIMARY KEY,
  ->     Nombre VARCHAR(100) NOT NULL,
  ->     Dirección VARCHAR(255) NOT NULL,
  ->     Población VARCHAR(100) NOT NULL,
  ->     id_administrador INT,
  ->     FOREIGN KEY (id_administrador) REFERENCES Administradores(id_administrador)
  ->     ON DELETE CASCADE
  ->     ON UPDATE CASCADE
  -> );
Query OK, 0 rows affected (0.062 sec)

MySQL [pruebapractica]> CREATE TABLE Vecinos (
  ->     id_vecino INT AUTO_INCREMENT PRIMARY KEY,
  ->     Nombre VARCHAR(100) NOT NULL,
  ->     Escalera CHAR(1) NOT NULL,
  ->     Piso INT NOT NULL,
  ->     Puerta CHAR(1) NOT NULL,
  ->     id_comunidad INT,
  ->     FOREIGN KEY (id_comunidad) REFERENCES Comunidades(id_comunidad)
  ->     ON DELETE CASCADE
  ->     ON UPDATE CASCADE
  -> );
Query OK, 0 rows affected (0.038 sec)
```

2.- Realiza el pseudocódigo para dar de alta un vecino.

Modelo

Clase VecinoModelo:

Método crearVecino(nombre, escalera, piso, puerta, id_comunidad):

SQL = "INSERT INTO Vecinos (Nombre, Escalera, Piso, Puerta, id_comunidad)
VALUES (?, ?, ?, ?, ?)"

Parámetros = (nombre, escalera, piso, puerta, id_comunidad)

EjecutarSQL(SQL, Parámetros)

Retornar resultado de la operación

Vista

Clase VecinoVista:

Método mostrarFormularioAltaVecino():

Mostrar "Formulario Alta Vecino"

Mostrar campo "Nombre"

Mostrar campo "Escalera"

Mostrar campo "Piso"

Mostrar campo "Puerta"
Mostrar campo "ID Comunidad"
Mostrar botón "Enviar"

Método obtenerDatosFormulario():
 nombre = Leer valor del campo "Nombre"
 escalera = Leer valor del campo "Escalera"
 piso = Leer valor del campo "Piso"
 puerta = Leer valor del campo "Puerta"
 id_comunidad = Leer valor del campo "ID Comunidad"
 Retornar (nombre, escalera, piso, puerta, id_comunidad)

Método mostrarMensaje(mensaje):
 Mostrar mensaje

Controlador

Clase VecinoControlador:

Método __init__(modelo, vista):
 self.modelo = modelo
 self.vista = vista

Método altaVecino():
 vista.mostrarFormularioAltaVecino()
 datos = vista.obtenerDatosFormulario()
 resultado = modelo.crearVecino(datos.nombre, datos.escalera, datos.piso,
datos.puerta, datos.id_comunidad)
 Si resultado es exitoso:
 vista.mostrarMensaje("Vecino dado de alta exitosamente.")
 De lo contrario:
 vista.mostrarMensaje("Error al dar de alta al vecino.")

3.- Desarrolla mediante el modelo MVC y utilizando PHP con POO la parte del programa que sirva para añadir una comunidad a la Base de Datos.

Añadir Nueva Comunidad

Nombre:

Dirección:

Población:

ID Administrador:

The screenshot shows a database management interface. At the top, there's a toolbar with various icons. Below it, a SQL query is entered: `1 • SELECT * FROM pruebapractica.comunidades;`. The results are displayed in a table with the following columns: `id_comunidad`, `Nombre`, `Dirección`, `Población`, and `id_administrador`. The table contains two rows: one with values 4, Casona, Calle del..., Madrid, 1, and another row with NULL values. The interface also includes a 'Filter Rows' section, an 'Edit' button, and an 'Export/Import' button. At the bottom, there's a status bar showing 'comunidades 1' and buttons for 'Apply' and 'Revert'.

	id_comunidad	Nombre	Dirección	Población	id_administrador
▶	4	Casona	Calle del...	Madrid	1
*	NULL	NULL	NULL	NULL	NULL

4.- Crea el proyecto en GitHub y sube los archivos. Indica la dirección del GitHub.

<https://github.com/dianacastano/comunidad>

5.- Crea un proyecto Laravel llamado pruebapráctica y crea las migraciones y vistas para la tabla Vecinos. (En este apartado no es necesario subir a github el código. Con realizar las capturas de pantalla suficientes para demostrar lo que se ha realizado del ejercicio y los comandos utilizados sería suficiente).

```
C:\xampp\htdocs>composer create-project --prefer-dist laravel/laravel="9.*" pruebapra  
ctica  
Creating a "laravel/laravel=9.*" project at "./pruebapractica"
```

```
C:\xampp\htdocs\pruebapractica>php artisan make:migration create_vecinos_table

INFO Migration [C:\xampp\htdocs\pruebapractica\database\Migrations\2024_07_30_165802_create_vecinos_table.php] created successfully.
```

```
C:\xampp\htdocs\pruebapractica>php artisan migrate

INFO Running migrations.

2024_07_30_165802_create_vecinos_table ..... 6ms DONE
```

```
C:\xampp\htdocs\pruebapractica>php artisan migrate:status

Migration name ..... Batch / Status
2019_12_14_000001_create_personal_access_tokens_table ..... [1] Ran
2024_07_30_165802_create_vecinos_table ..... [2] Ran
```

```
C:\xampp\htdocs\pruebapractica>php artisan make:model Vecino

INFO Model [C:\xampp\htdocs\pruebapractica\app\Models\Vecino.php] created successfully.
```

```
C:\xampp\htdocs\pruebapractica>php artisan make:controller VecinoController

INFO Controller [C:\xampp\htdocs\pruebapractica\app\Http\Controllers\VecinoController.php] created successfully.
```

Añadir Vecino

Nombre:

Escalera:

Piso:

Puerta:

ID Comunidad: