Taller de Lenguajes II – 2024
Programador Universitario / Licenciatura en informática / Ingeniería en Informática
TP Nro 5

CONTENIDOS

- Base de Datos SQLite
- Sentencias SQL
- ADO Net
- Repositorios

INTRODUCCIÓN

Copie el siguiente enlace en su navegador: https://classroom.github.com/a/DXlHZqYf esto creará el repositorio para poder subir el **Trabajo Práctico Nro. 5**.

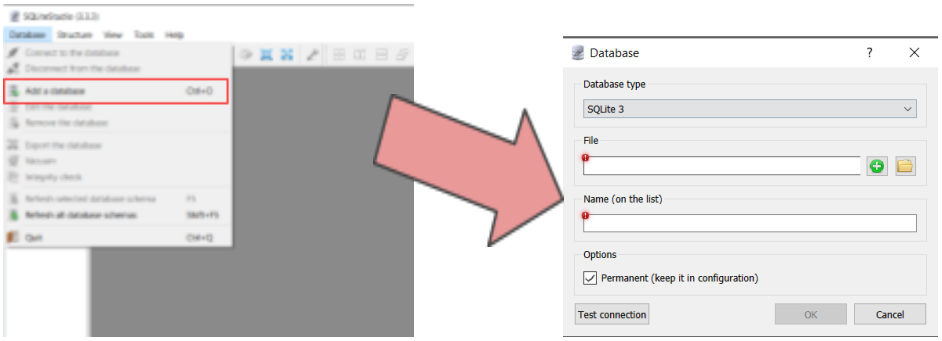
Se nos solicita implementar un prototipo de generador de presupuestos de productos que será utilizado como prototipo para la página web de una Distribuidora de Insumos Informáticos.

1) Descargue la aplicación SQLite Studio (https://sqlitestudio.pl),. Este software es una IDE que le permitirá administrar bases de datos SQLite de forma sencilla. Su interfaz gráfica facilita la navegación, visualización y modificación de datos dentro de las bases de datos SQLite.
2) Una vez que haya descargado e instalado SQLite Studio, continúe con la apertura de la base de datos proporcionada. El proyecto que clonó incluye una base de datos llamada **Tienda.db**, que ya contiene las siguientes tablas:

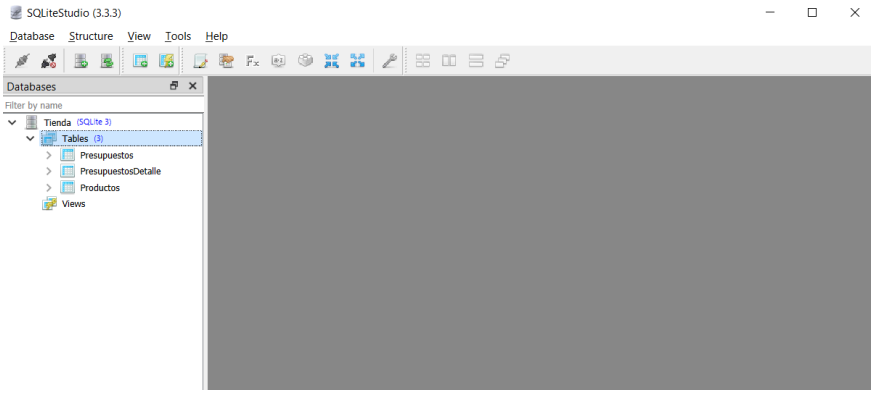
    a) Productos
        i) *idProducto
        ii) Descripcion
        iii) Precio
    b) Presupuestos
        i) *IdPresupuesto
        ii) NombreDestinatario
        iii) FechaCreacion
    c) PresupuestosDetalle
        i) *idPresupuesto
        ii) *idProducto
        iii) Cantidad

Abrir y explorar la base de datos

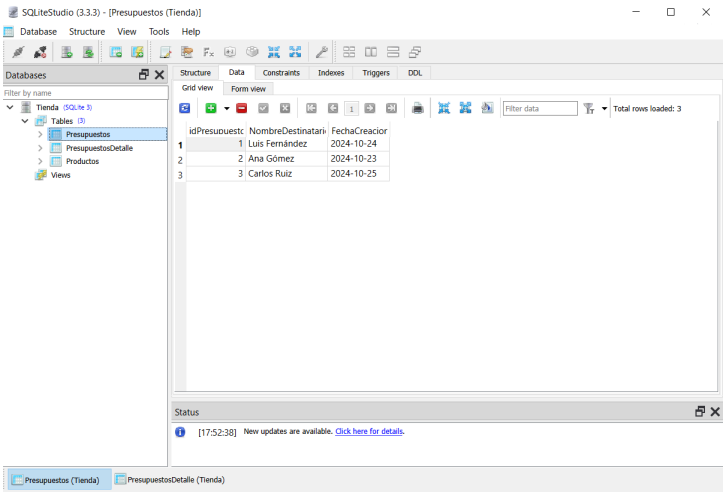
1. Abra **SQLite Studio** y en la barra de herramientas superior, seleccione la opción “Add database” o “Agregar Base de datos” (siga la **Imagenes** para orientación).



2. Navegue hasta la ubicación de **Tienda.db** y ábrala. Al cargar la base de datos, verá las tablas **Productos**, **Presupuestos** y **PresupuestosDetalle** en el panel izquierdo de la ventana (deberia tener algo similar a lo que muestra la **Imagen 2**).



3. Haga clic en cada tabla para visualizar los datos almacenados y navegar entre ellos. Por ejemplo, puede ver los productos existentes, los presupuestos generados y los detalles de cada presupuesto, como se muestra en la **Imagen 3**.



3) **En SQLite Studio pruebe las consultas SQL y vea los resultados.**

   a) **Insertar un Producto**

   INSERT INTO Productos (Descripcion, Precio) VALUES ('[Descripcion]', [Precio]);

   Ej:

   INSERT INTO Productos (Descripcion, Precio) VALUES ('Mouse Inalámbrico Logitech', 5000.0);

   b) **Insertar un Presupuesto**

   INSERT INTO Presupuestos (NombreDestinatario, FechaCreacion) VALUES ('[NombreDestinatario]', '[FechaCreacion]');

Ej:

INSERT INTO Presupuestos (NombreDestinatario, FechaCreacion) VALUES ('Carlos Ruiz', '2024-10-25');

c) Insertar registros en PresupuestoDetalle

INSERT INTO PresupuestosDetalle (idPresupuesto, idProducto, Cantidad) VALUES ([idPresupuesto], [idProducto], [Cantidad]);

Ej:

INSERT INTO PresupuestosDetalle (idPresupuesto, idProducto, Cantidad) VALUES (1, 3, 2); -- 2 unidades del producto con id 3 en el presupuesto 1

d) Modificar un producto

UPDATE Productos SET Descripcion = '[NuevaDescripcion]', Precio = [NuevoPrecio] WHERE idProducto = [idProducto];

Ej:

```
UPDATE Productos
SET Descripcion = 'Teclado Mecánico Logitech',
Precio = 12000
WHERE idProducto = 3;
```

e) Modificar NombreDestinatario de Presupuesto

UPDATE Presupuestos SET NombreDestinatario = '[NuevoNombreDestinatario]' WHERE idPresupuesto = [idPresupuesto];

Ej:

UPDATE Presupuestos SET NombreDestinatario = 'Luis Fernández' WHERE idPresupuesto = 1;

f) Eliminar un registro de PresupuestoDetalle

DELETE FROM PresupuestosDetalle WHERE idPresupuesto = [idPresupuesto] AND idProducto = [idProducto];

Ej:

DELETE FROM PresupuestosDetalle WHERE idPresupuesto = 1 AND idProducto = 2;

1) **Cree un nuevo proyecto Web Api**

- Para ello use los siguientes comando
  - **(versión 7): dotnet new webapi**
  - **(versión 8+): dotnet new webapi --use-controllers**
- Instale dependencia para utilizar SQLite con ADO.Net, para ello ejecute el siguiente comando:
  - **dotnet add package Microsoft.Data.SQLite**

2) **Creando las clases modelo:**

Crear una carpeta llamada Models donde ubicará los modelos de su aplicación. A continuación se detallan los modelos que deben crear:

- **Productos**
  - int idProducto
  - string descripcion
  - int precio
- **Presupuestos**
  - int IdPresupuesto
  - string nombreDestinatario
  - List<PresupuestoDetalle> detalle
  - Metodos
    - MontoPresupuesto ()
    - MontoPresupuestoConIva()
    - CantidadProductos ()
- **PresupuestosDetalle**
  - Productos producto
  - int cantidad

3) **Creando repositorios:**

Para una mejor organización y separación de responsabilidades en su API en .NET, es recomendable utilizar el Patrones de Repositorio para gestionar las operaciones de acceso a la base de datos. Cree una carpeta llamada *Repositorios*, donde ubicará los repositorios a construir.

A continuación se detallan los repositorios y los métodos que deben crear:

Repositorio de Productos:

Crear un repositorio llamado ProductoRepository para gestionar todas las operaciones relacionadas con Productos. Este repositorio debe incluir métodos para:

- Crear un nuevo Producto. (recibe un objeto Producto)
- Modificar un Producto existente. (recibe un Id y un objeto Producto)
- Listar todos los Productos registrados. (devuelve un List de Producto)
- Obtener detalles de un Productos por su ID. (recibe un Id y devuelve un Producto)
- Eliminar un Producto por ID

Repositorio de Presupuestos:

Crear un repositorio llamado PresupuestosRepository para gestionar todas las operaciones relacionadas con Presupuestos. Este repositorio debe incluir métodos para:

- Crear un nuevo Presupuesto. (recibe un objeto Presupuesto)
- Listar todos los Presupuestos registrados. (devuelve un List de Presupuestos)
- Obtener detalles de un Presupuesto por su ID. (recibe un Id y devuelve un Presupuesto con sus productos y cantidades)
- Agregar un producto y una cantidad a un presupuesto (recibe un Id)
- Eliminar un Presupuesto por ID

Taller de Lenguajes II – 2024
Programador Universitario / Licenciatura en informática / Ingeniería en Informática
TP Nro 5

4) Creando EndPoints:

A continuación, vamos a detallar los controladores y los distintos endpoints que se deberán agregar al sistema:

Crear un Controlador de Producto(ProductoController) que incluya los endpoints para:

- **POST** /api/Producto: Permite crear un nuevo Producto.
- **GET** /api/Producto: Permite listar los Productos existentes.
- **PUT** /api/Producto/{Id}: Permite modificar un nombre de un Producto.

Crear un Controlador de Presupuestos (PresupuestosController) que incluya los endpoints para:

- **POST** /api/Presupuesto: Permite crear un Presupuesto.
- **POST** /api/Presupuesto/{id}/ProductoDetalle: Permite agregar un Producto existente y una cantidad al presupuesto.
- **GET** /api/presupuesto: Permite listar los presupuestos existentes.
- **GET** /api/Presupuesto/{id}: Permite agregar un Producto existente y una cantidad al presupuesto.

Nota:

La Estructura del Proyecto tendría que quedar de la siguiente manera

- /Models
  - Productos.cs
  - Presupuestos.cs
  - PresupuestosDetalle.cs
- /Repositorios
  - ProductosRepository.cs
  - PresupuestosRepository.cs
- /Controllers
  - ProductosController.cs
  - PresupuestosController.cs