UTN
Facultad Regional Córdoba

TUP

Tecnicatura Universitaria en Programación

PROGRAMACIÓN II

Unidad Temática N°4:
Introducción a Servicios

Guía de Estudio
1° Año – 2° Cuatrimestre

Índice

WEB APIs

Problema 4.1:

Crear proyecto Web API sobre .Net 5.0 utilizando IDE Visual Studio. La API debe exponer un único método GET que retorne la fecha actual: número, día de la semana, mes y año.

- Definir un modelo: Fecha con las datos antes mencionados
- Probar la API mediante la herramienta integrada Swagger.

Problema 4.2:

Crear proyecto Web API sobre .Net 5.0 utilizando IDE Visual Studio. La API debe contener:

- Un controlador llamado CashController que tenga como atributo de clase una lista de objetos **Moneda**. Cada moneda contiene: Nombre y valor en pesos de dicha moneda. Por ejemplo: Nombre: {“Dolar”, Valor:180}, {“Peso argentino”, Valor:1}
- 1 método GET: que permita obtener todos los objetos **Moneda** creados hasta el momento.
- 1 método GET/1: con un parámetro que sea el nombre de la moneda a consultar. Si no la encuentra deberá informar con un mensaje: “Moneda no registrada”
- 1 método POST que permita crear una moneda y agregarla a la lista. Como respuesta este método devuelve el objeto creado.
- Probar la API utilizando POSTMAN.

Problema 4.3:

Crear proyecto Web API sobre .Net 5.0 utilizando IDE Visual Studio. La API debe contener:

- Un controlador llamado TemperaturaController.
- Una implementación de un patrón Singleton que exponga los comportamientos de una un único objeto de tipo RegistroTemp que represente el repositorio de temperaturas registradas.
- De cada temperatura se registran los datos: identificador IOT (int), fechaHora (datetime) y valor (float).

- 1 método GET: que permita obtener todas las lecturas de temperatura registradas. En caso de haber lecturas registradas, devolver una lista vacía.
- 1 método GET/1: que permita obtener todas las lecturas de temperatura enviadas por un dispositivo en particular (se recibe el número de identificación IOT). En caso de haber lecturas registradas, devolver una lista vacía.
- 1 método POST que permita registrar los datos de una temperatura. Se devuelve siempre un mensaje de confirmación.
- Probar la API utilizando POSTMAN.

Problema 4.4:

Crear proyecto Web API sobre .Net 5.0 utilizando IDE Visual Studio. La API debe contener:

- Un controlador llamado ProductosController.
- Una implementación de una interfaz IAplicacion que exponga los servicios para: agregar, consultar, editar y borrar productos. La Aplicación tiene una Lista de productos en memoria.
- De cada producto se conocen los siguientes datos: código, nombre y precio.
- Métodos para hacer un CRUD de productos.
- Probar la API utilizando POSTMAN.

Problema 4.5:

Modificar el ejercicio 4.4 para que la gestión CRUD de productos se realice sobre una base de datos: db_empresa con el siguiente script de creación:

```
USE [db_empresa]
CREATE TABLE [dbo].[PRODUCTOS](
	[codigo] [int] IDENTITY(1,1) NOT NULL,
	[nombre] [nchar](50) NOT NULL,
	[precio] [real] NOT NULL,
 CONSTRAINT [PK_PRODUCTOS] PRIMARY KEY CLUSTERED
(
	[codigo] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Problema 4.6:

Crear proyecto Web API sobre .Net 5.0 utilizando IDE Visual Studio. La API debe contener:

- Un controlador llamado RecetaController.
- Una implementación de una interfaz IAplicacion que exponga los servicios para: agregar, consultar, editar y borrar recetas junto con el detalle de ingredientes que la componen.
- Modelar una capa de acceso a datos que permita, mediante procedimientos almacenados, gestionar Recetas
- La base de datos se llama recetas_db y tiene el siguiente script de creación:

```
USE [recetas_db]
CREATE TABLE [dbo].[Recetas](
[id_receta] [int] NOT NULL,
[nombre] [varchar](50) NOT NULL,
[cheff] [varchar](100) NULL,
 CONSTRAINT [PK_Recetas] PRIMARY KEY CLUSTERED
(
[id_receta] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]

CREATE TABLE [dbo].[Ingredientes](
[id_ingrediente] [int] NOT NULL,
[n_ingrediente] [varchar](50) NOT NULL,
[cantidad] [float] NOT NULL,
[id_receta] [int] NOT NULL,
 CONSTRAINT [PK_Ingredientes] PRIMARY KEY CLUSTERED
(
[id_ingrediente] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON)
ON [PRIMARY]
) ON [PRIMARY]
GO
SET ANSI_PADDING OFF
GO
/****** Object:  ForeignKey [FK_Ingredientes_Recetas]    Script
Date: 09/27/2021 01:27:54 ******/
ALTER TABLE [dbo].[Ingredientes]  WITH CHECK ADD  CONSTRAINT
[FK_Ingredientes_Recetas] FOREIGN KEY([id_receta])
REFERENCES [dbo].[Recetas] ([id_receta])
GO
ALTER TABLE [dbo].[Ingredientes] CHECK CONSTRAINT
[FK_Ingredientes_Recetas]
```

- Cada objeto Receta tiene: código, nombre, cheff (creador) y un lista de ingredientes.
- Cada objeto Ingrediente tiene: código, nombre y cantidad
- Métodos para hacer un CRUD de recetas.
- Probar la API utilizando POSTMAN.