UTN
Facultad Regional Córdoba

TUP

Tecnicatura Universitaria en Programación

PROGRAMACIÓN II

Unidad Temática N°2:

Arquitectura cliente servidor y Servicios Web

Guía de Estudio

1° Año – 2° Cuatrimestre

Índice

WEB APIs

Problema 2.1:

Crear proyecto Web API sobre .Net 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 los datos antes mencionados
- Probar la API mediante la herramienta integrada Swagger.

Problema 2.2:

Crear proyecto Web API sobre .Net 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 2.3:

Crear proyecto Web API sobre .Net 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 2.4:

Crear proyecto Web API sobre .Net 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 2.5:

Refactorizar la solución del **problema 2.4** para que la gestión CRUD de productos se realice sobre una base de datos: db_productos con el siguiente script de creación:

```
USE [db_productos]
CREATE TABLE [dbo].[PRODUCTOS](
	[codigo] [int] IDENTITY(1,1) NOT NULL,
	[nombre] [nchar](50) NOT NULL,
	[precio] [real] NOT NULL,
	[fecha_baja] [datetime] NULL,
	[motivo_baja][nchar](50) 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]
```

- Modelar una capa de datos utilizando los patrones vistos en la unidad.
- Deberá validar las siguientes reglas de negocio:
  - ✓ No se pueden registrar dos productos con el mismo nombre
  - ✓ Solo es posible actualizar/borrar productos sin datos de baja (fecha y motivo de baja nulos)

Problema 2.6:

Crear proyecto Web API sobre .Net 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. Implementar el patrón Repository.
- Crear los procedimientos almacenados en la base de datos
- 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]
```

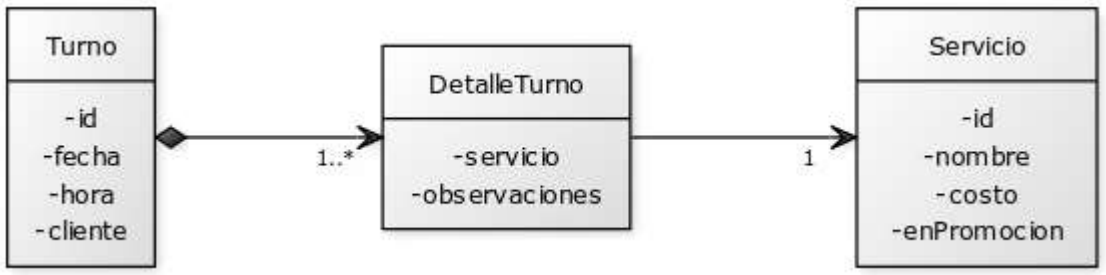
```
GO
SET ANSI_PADDING OFF
GO
/****** Object:  ForeignKey [FK_Ingredientes_Recetas]    Script
Date: 09/27/2021 01:27:54 ******/
ALTER TABLE [dbo].[Ingredientes]  WITH CHECK ADD  CONSTRAINT
[FK_Ingredientes_Recetas] FOREIGN KEY([id_receta])
REFERENCES [dbo].[Recetas] ([id_receta])
GO
ALTER TABLE [dbo].[Ingredientes] CHECK CONSTRAINT
[FK_Ingredientes_Recetas]
```

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

Problema 2.7:

Se necesita desarrollar una Web API que permita gestionar los turnos de una peluquería. Se pide:

- Crear las clases necesarias del modelo de dominio que se muestra en la siguiente figura:



- Definir una capa de datos que permita consultar servicios y registrar los datos completos de un turno con el detalle de servicios. Implementar el patrón **repository** tanto para turnos como para servicios.
- Cada repositorio deberá llamar a los procedimientos almacenados provistos en el script.
- Adicionalmente considerar las siguientes reglas de negocio:
  - ✓ La fecha de reserva deberá tener como valor por defecto la fecha actual + 1 (fecha día siguiente como mínimo). Deberá controlar que la fecha de reserva no supere los 45 días a la fecha actual.

- ✓ Deberá controlar que no se pueden grabar dos veces el mismo servicio como detalle. Es decir, no puede solicitar “corte de cabello” 2 veces en el mismo turno.
- ✓ Deberá controlar además que solo es posible registrar el turno si para la fecha y hora seleccionadas no existe un registro previamente cargado.
- ✓ Controlar que se hayan ingresado datos de al menos un servicio.
- ✓ Al registrar un turno se deberá retornar objeto mensaje de confirmación.

- Crear una colección POSTMAN que permita probar las funcionalidades para consultar servicios y registrar los datos completos de un turno

Script de creación e inicialización:

```
USE [db_turnos]
GO
/****** Object:  Table [dbo].[T_TURNOS]    Script Date: 02/05/2024
14:05:32 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[T_TURNOS](
[id] [int] IDENTITY(1,1) NOT NULL,
[fecha] [varchar](10) NULL,
[hora] [varchar](5) NULL,
[cliente] [varchar](100) NULL,
 CONSTRAINT [PK_T_TURNOS] PRIMARY KEY CLUSTERED
(
[id] 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:  Table [dbo].[T_SERVICIOS]    Script Date:
02/05/2024 14:05:32 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[T_SERVICIOS](
[id] [int] NOT NULL,
[nombre] [varchar](50) NOT NULL,
[costo] [int] NOT NULL,
[enPromocion] [varchar](1) NOT NULL,
 CONSTRAINT [PK_T_SERVICIOS] PRIMARY KEY CLUSTERED
(
[id] 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
INSERT [dbo].[T_SERVICIOS] ([id], [nombre], [costo], [enPromocion])
VALUES (1, N'Lavado de cabello', 500, N'N')
INSERT [dbo].[T_SERVICIOS] ([id], [nombre], [costo], [enPromocion])
VALUES (2, N'Corte', 2000, N'S')
INSERT [dbo].[T_SERVICIOS] ([id], [nombre], [costo], [enPromocion])
VALUES (3, N'Tintura', 3500, N'N')
INSERT [dbo].[T_SERVICIOS] ([id], [nombre], [costo], [enPromocion])
VALUES (4, N'Alisado', 12000, N'N')
INSERT [dbo].[T_SERVICIOS] ([id], [nombre], [costo], [enPromocion])
VALUES (5, N'Nutrición', 12500, N'S')
INSERT [dbo].[T_SERVICIOS] ([id], [nombre], [costo], [enPromocion])
VALUES (6, N'Tratamiento de Keratina', 20000, N'N')
/****** Object:  Table [dbo].[T_DETALLES_TURNO]    Script Date:
02/05/2024 14:05:32 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
CREATE TABLE [dbo].[T_DETALLES_TURNO](
[id_turno] [int] NOT NULL,
[id_servicio] [int] NOT NULL,
[observaciones] [varchar](200) NULL,
 CONSTRAINT [PK_T_DETALLES_TURNO] PRIMARY KEY CLUSTERED
(
[id_turno] ASC,
[id_servicio] 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:  StoredProcedure [dbo].[SP_INSERTAR_DETALLES]
Script Date: 02/05/2024 14:05:30 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SP_INSERTAR_DETALLES]
@id_turno int,
@id_servicio int,
@observaciones varchar(200)
AS
BEGIN
INSERT INTO T_DETALLES_TURNO(id_turno,id_servicio, observaciones)
    VALUES (@id_turno,@id_servicio, @observaciones);

END
GO
/****** Object:  StoredProcedure [dbo].[SP_CONTAR_TURNOS]    Script
Date: 02/05/2024 14:05:30 ******/
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SP_CONTAR_TURNOS]
    @fecha VARCHAR(10),
    @hora VARCHAR(8),
    @ctd_turnos INT OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    SELECT @ctd_turnos = COUNT(*)
    FROM T_TURNOS
    WHERE fecha = @fecha AND hora = @hora;
END;
GO
/****** Object:  StoredProcedure [dbo].[SP_CONSULTAR_SERVICIOS]
Script Date: 02/05/2024 14:05:30 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[SP_CONSULTAR_SERVICIOS]
AS
BEGIN

SELECT * from T_SERVICIOS ORDER BY 2;
END
GO
```

Problema 2.8:

Tomando como base el **Problema 2.7** refactorizar la solución para que la Web API exponga servicios para:

- Actualizar los datos de una reserva siempre que la fecha/hora sean anteriores a los confirmados en su creación
- Cancelar una reserva indicando un motivo de cancelación. Al igual que el apartado anterior, validar la restricción temporal.
- Actualizar el modelo de datos con los campos: fecha_cancelación y motivo_cancelación en la tabla reservas.
- Actualizar la colección POSTMAN para probar los escenarios de actualización y cancelación de reservas.