**федеральное государственное автономное образовательное учреждение высшего образования**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**(ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ)**

**(Факультет информационных технологий)**

***(Институт Принтмедиа и информационных технологий)***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

**Дисциплина: BackEnd-разработка**

### Тема: Создание консольного приложения с внедренными зависимостями на основе ASP.NET Core

**Выполнил(а): студент(ка) группы 221-3711**

Ежов Тимофей Алексеевич

(Фамилия И.О.)

**Дата, подпись** 25.02.2024

**Проверил:**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

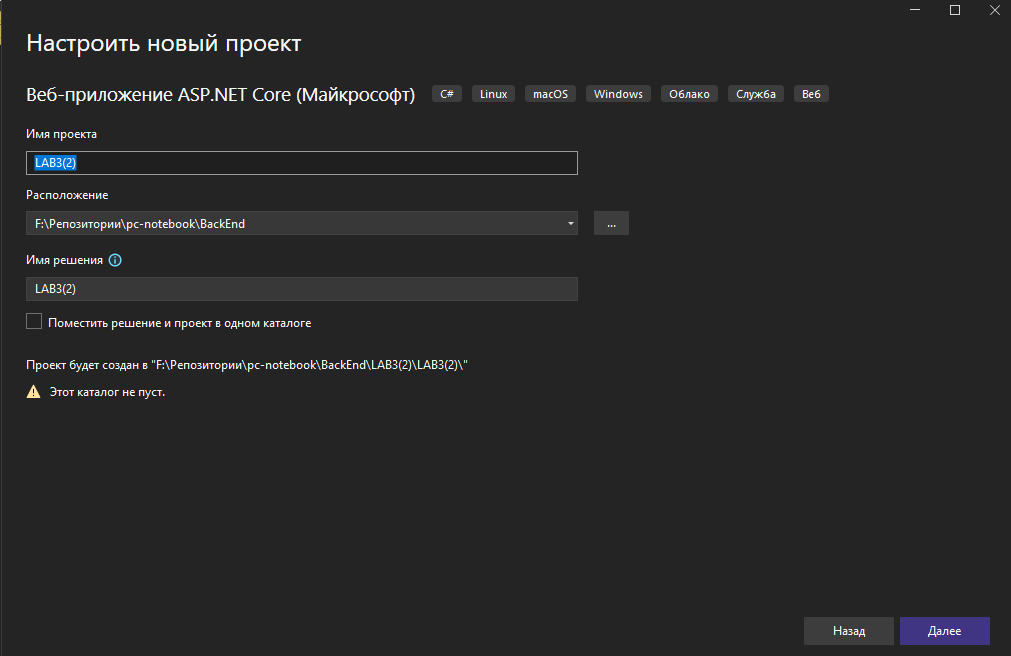
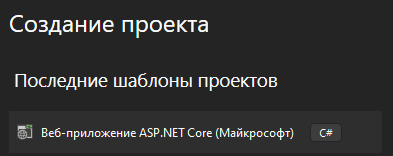
**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

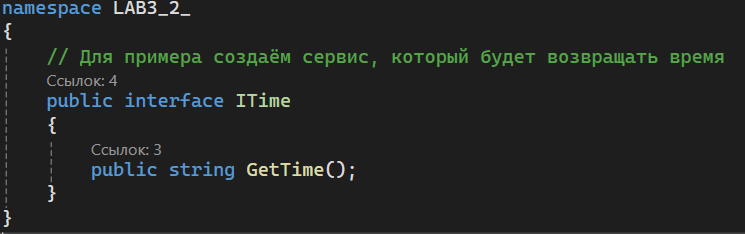
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

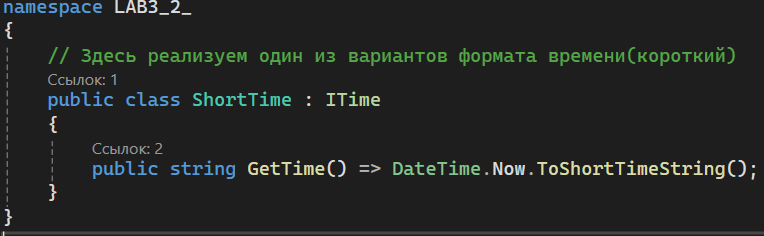
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

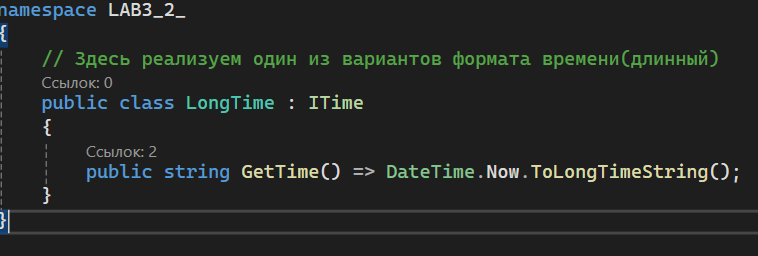
**Москва2024**

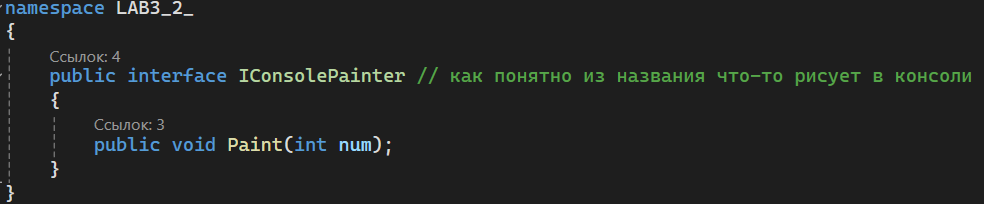
Этапы:

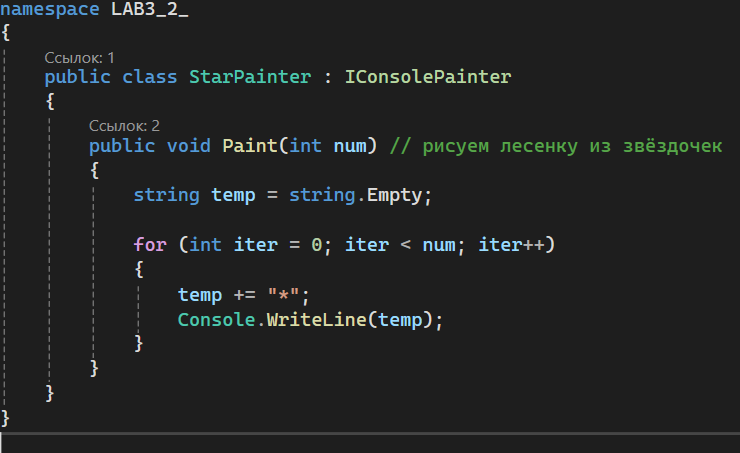
1. Создание пустого проекта из шаблона веб приложение ASP.Net core в Visual Studio
2. Создаём Какой-то функционал для приложения



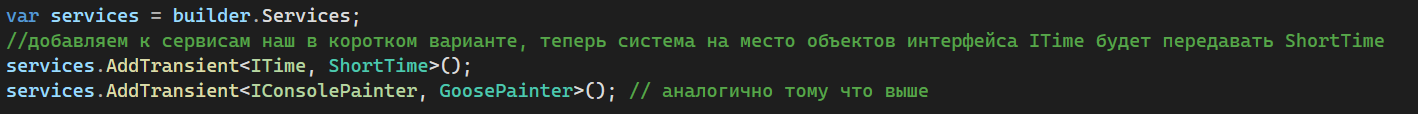
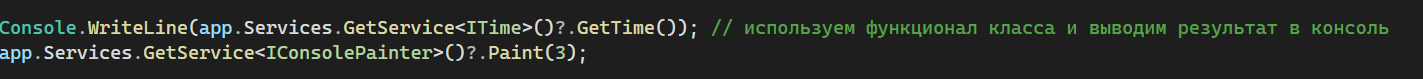


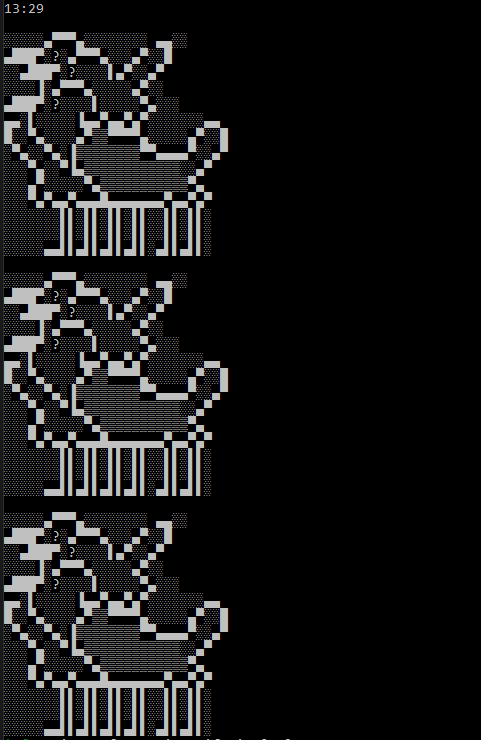






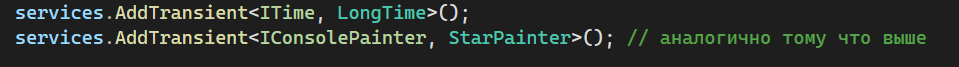


1. Добавим наш прекрасный функционал к сервисам, выбрав варианты реализации (указав конкретный тип класса)
2. Добавим вывод в консоль
3. Поверим работу



Всё работает корректно

1. Теперь сменим класс который передаём в сервисы



1. Запустим ещё раз



Работа корректна

Выводы:

Зависимости удобно использовать, если планируется расширение функционала приложения.

# Листинг

using LAB3\_2\_;

var builder = WebApplication.CreateBuilder(args);

var services = builder.Services;

//добавляем к сервисам наш в коротком варианте, теперь система на место объектов интерфейса ITime будет передавать ShortTime

services.AddTransient<ITime, LongTime>();

services.AddTransient<IConsolePainter, StarPainter>(); // аналогично тому что выше

var app = builder.Build();

Console.WriteLine(app.Services.GetService<ITime>()?.GetTime()); // используем функционал класса и выводим результат в консоль

app.Services.GetService<IConsolePainter>()?.Paint(3);

app.Run();

namespace LAB3\_2\_

{

    // Для примера создаём сервис, который будет возвращать время

    public interface ITime

    {

        public string GetTime();

    }

}

namespace LAB3\_2\_

{

    // Здесь реализуем один из вариантов формата времени(короткий)

    public class ShortTime : ITime

    {

        public string GetTime() => DateTime.Now.ToShortTimeString();

    }

}

namespace LAB3\_2\_

{

    // Здесь реализуем один из вариантов формата времени(длинный)

    public class LongTime : ITime

    {

        public string GetTime() => DateTime.Now.ToLongTimeString();

    }

}

namespace LAB3\_2\_

{

    public interface IConsolePainter // как понятно из названия что-то рисует в консоли

    {

        public void Paint(int num);

    }

}

namespace LAB3\_2\_

{

    public class StarPainter : IConsolePainter

    {

        public void Paint(int num) // рисуем лесенку из звёздочек

        {

            string temp = string.Empty;

            for (int iter = 0; iter < num; iter++)

            {

                temp += "\*";

                Console.WriteLine(temp);

            }

        }

    }

}

namespace LAB3\_2\_

{

    public class GoosePainter : IConsolePainter

    {

        public void Paint(int num) // рисуем гуся-гидру num раз

        {

            for (int iter = 0; iter < num; iter++)

            {

                Console.WriteLine();

                Console.Write("░░░░░▄▀▀▀▄░░░░░░░░ ▄▄░░\r\n▄███▀░◐░▄▀▀▀▄░░░▄▀░░█\r\n░░▄███▀░◐░░░░▌▄▀░░▄▀\r\n" +

                    "░░░░▐░▄▀▀▀▄░░░░░▄▀░░" +

                    "\r\n▄███▀░◐░░░░▌░░░░░▀▄░░░\r\n▄▄░▌░░░░░▐▄▄▀▄▄▀▄▀░░░░░░░▄▄\r\n█░░▀▄░░░░▄▀▒▒▀▀▀▀▄░░░░░▄▀" +

                    "░░█\r\n░▀▄░░▀▄░▐▒▒▒▒▒▒▒▒▀▀▄▄▄▄▀░░▄▀\r\n" +

                    "░░░▀▄░░▀▐▄▒▒▒▒▒▒▒▒▒▒▒▒░░▄▀\r\n░░░▄▀░░░░░▀▄▒▒▒▒▒▒▒▒▒▒▒▀▄\r\n░░░▀▄▀▄▄▀▄▄▄█▄▄▄▄▄▄▄▀▄▄▀▄▀" +

                    "\r\n░░░░░░░▌▌░▌▌░▌▌░▌▌░░▌▌░▌▌░\r\n░░░░░░░▌▌░▌▌░▌▌░▌▌░░▌▌░▌▌░" +

                    "\r\n░░░░░▄▄▌▌▄▌▌▄▌▌▄▌▌░▄▌▌▄▌▌░");

                Console.WriteLine();

            }

        }

    }

}