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: _____

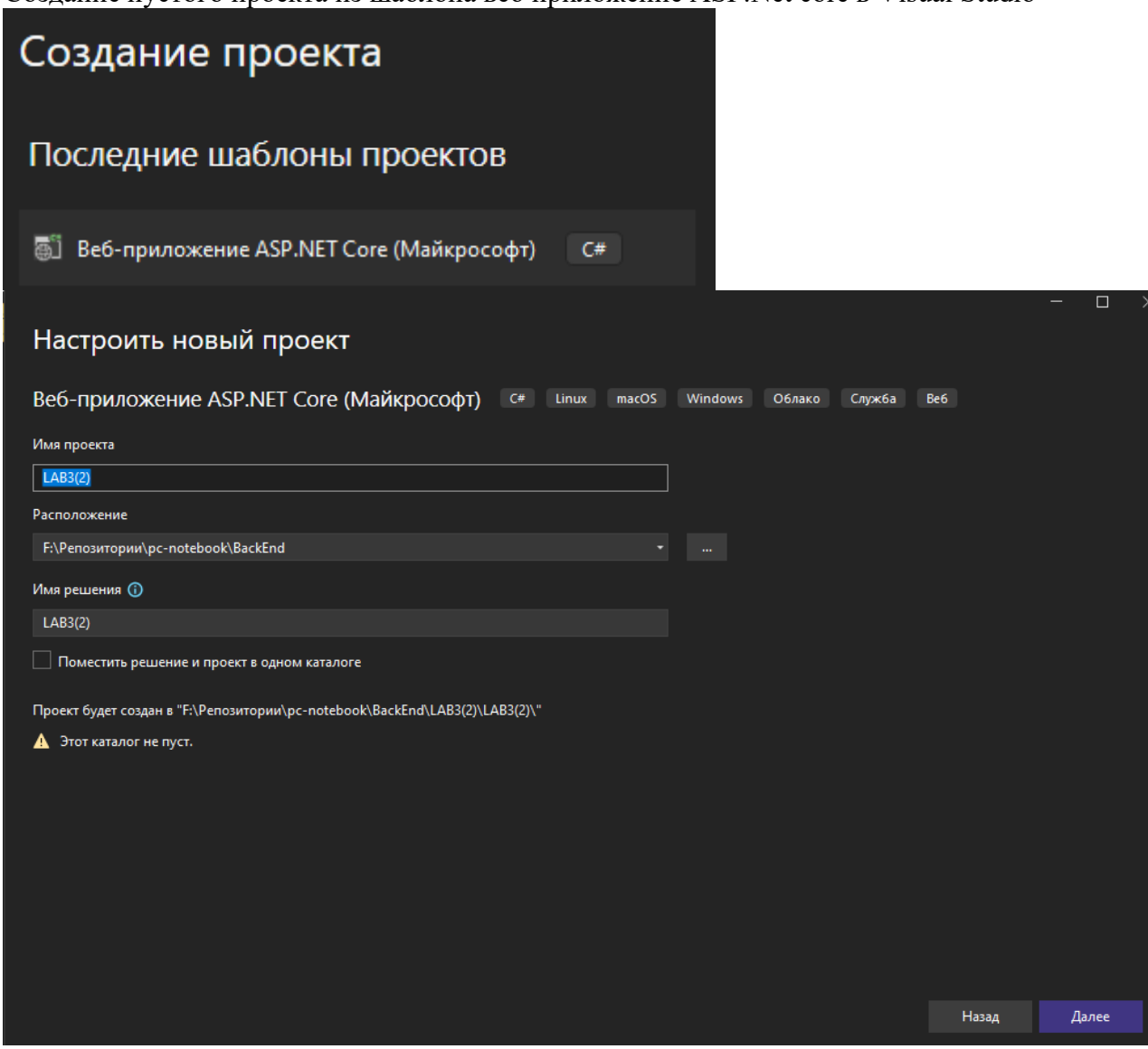
_____

_____

Москва 2024

Этапы:

1) Создание пустого проекта из шаблона веб приложение ASP.Net core в Visual Studio



2) Создаём Какой-то функционал для приложения

```
namespace LAB3_2_
{
    // Для примера создаём сервис, который будет возвращать время
    Ссылок: 4
    public interface ITime
    {
        Ссылок: 3
        public string GetTime();
    }
}
```

```
namespace LAB3_2_
{
    // Здесь реализуем один из вариантов формата времени(короткий)
    Ссылок: 1
    public class ShortTime : ITime
    {
        Ссылок: 2
        public string GetTime() => DateTime.Now.ToShortTimeString();
    }
}
```

```
namespace LAB3_2_
{
    // Здесь реализуем один из вариантов формата времени(длинный)
    Ссылок: 0
    public class LongTime : ITime
    {
        Ссылок: 2
        public string GetTime() => DateTime.Now.ToLongTimeString();
    }
}
```

```
namespace LAB3_2_
{
    Ссылок: 4
    public interface IConsolePainter // как понятно из названия что-то рисует в консоли
    {
        Ссылок: 3
        public void Paint(int num);
    }
}
namespace LAB3_2_
{
    Ссылок: 1
    public class StarPainter : IConsolePainter
    {
        Ссылок: 2
        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_
{
    Ссылок: 1
    public class GoosePainter : IConsolePainter
    {
        Ссылок: 2
        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
                Console.WriteLine();
            }
        }
    }
}
```

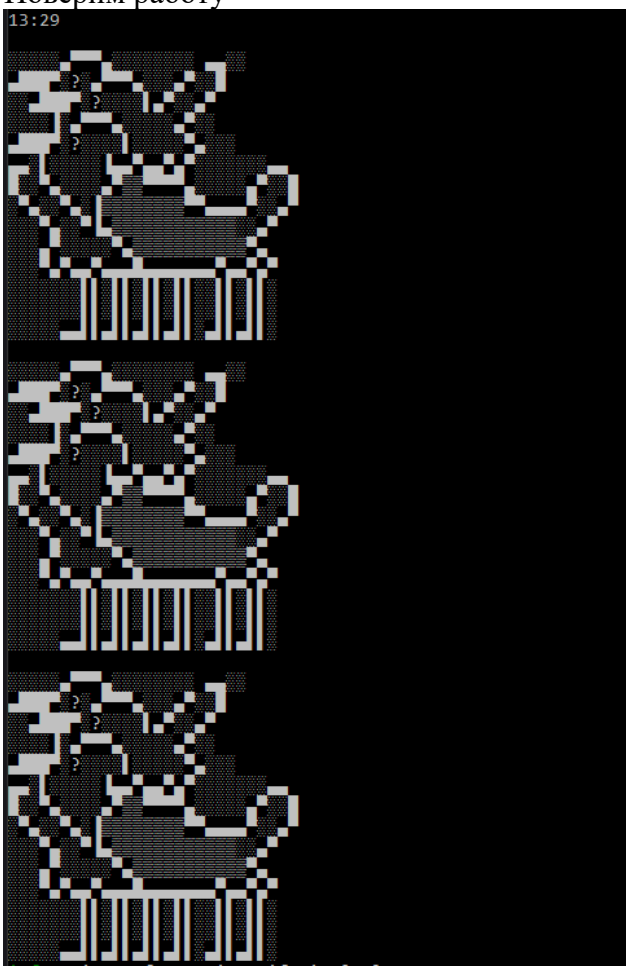
3) Добавим наш прекрасный функционал к сервисам, выбрав варианты реализации (указав конкретный тип класса)

```
var services = builder.Services;
//добавляем к сервисам наш в коротком варианте, теперь система на место объектов интерфейса ITime будет передавать ShortTime
services.AddTransient<ITime, ShortTime>();
services.AddTransient<IConsolePainter, GoosePainter>(); // аналогично тому что выше
```

4) Добавим вывод в консоль

```
Console.WriteLine(app.Services.GetService<ITime>()?.GetTime()); // используем функционал класса и выводим результат в консоль
app.Services.GetService<IConsolePainter>()?.Paint(3);
```

5) Поверим работу

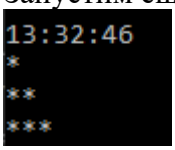


Всё работает корректно

6) Теперь сменим класс который передаём в сервисы

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

7) 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();
    }
}
```

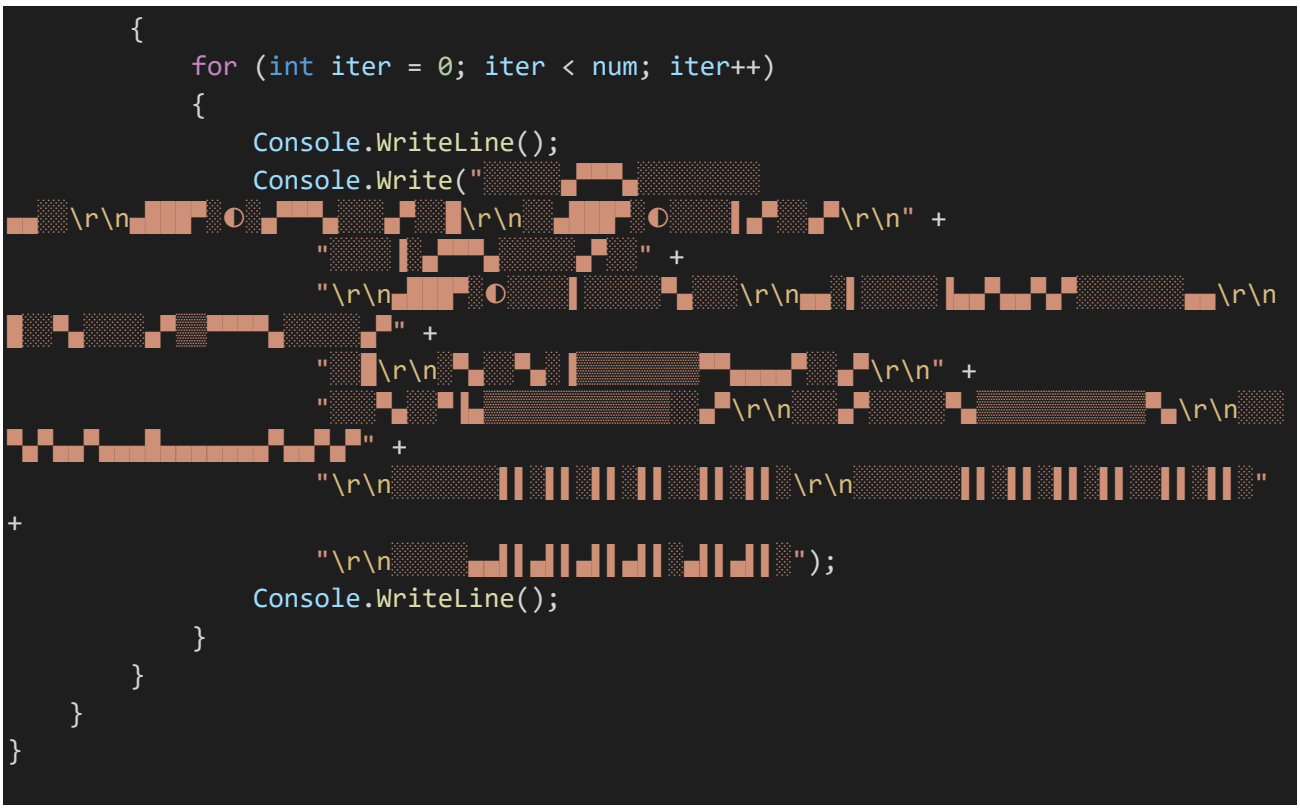
```
namespace LAB3_2_
{
    // Здесь реализуем один из вариантов формата времени(длинный)
    public class LongTime : ITime
    {
        public string GetTime() => DateTime.Now.ToLongTimeString();
    }
}
```

```
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
");
Console.WriteLine();
}
}
}
}