**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет 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: студент группы 231-332**

**Чесноков Александр Евгеньевич**

(Фамилия И.О. обучающегося)

**Дата, подпись** 01.06.2025 

(Дата)(Подпись)

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

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

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

(Дата)(Подпись)

**Москва**

**2025**

**Цель:**

Освоить создание консольного приложения, использующего внедрение зависимостей в ASP.NET Core.

**Ход работы:**

В рамках лабораторной работы было создано консольное приложение на платформе .NET с использованием механизма внедрения зависимостей (Dependency Injection) в ASP.NET Core. Основная структура программы начинается с настройки хоста с помощью метода Host.CreateDefaultBuilder, в процессе которой формируется контейнер зависимостей. В этот контейнер регистрируется служба уведомлений, реализующая интерфейс INotificationService, и основной класс приложения Application. Регистрация выполнена через метод AddSingleton, что означает создание единственного экземпляра объектов в течение работы программы.

Интерфейс INotificationService выступает абстракцией, определяющей контракт для службы уведомлений, а класс NotificationService реализует этот контракт, предоставляя метод для получения сообщения. Такой подход позволяет отделить определение поведения от его реализации, что упрощает замену и тестирование компонентов. В классе Application внедрение зависимости происходит через конструктор: объект INotificationService автоматически передается при создании экземпляра Application благодаря механизму Dependency Injection.

Запуск приложения осуществляется через получение зарегистрированного в контейнере объекта Application и вызов его метода Run. Внутри метода Run используется внедренный сервис для получения строки сообщения, которая выводится в консоль. Таким образом, ключевыми шагами стали настройка контейнера зависимостей, регистрация сервисов, реализация интерфейса для абстракции и использование внедренного сервиса внутри основного класса приложения. Это позволило на практике освоить основы работы с внедрением зависимостей в консольных приложениях на ASP.NET Core.

**Скриншоты работы приложения:**

****

**Листинг:**

**Program.cs:**

|  |
| --- |
| namespace lab2  {      public class Program      {          public static void Main(string[] args)          {              // *Создание и настройка хоста приложения*              var host = Host.CreateDefaultBuilder(args).ConfigureServices((hostContext, services) =>              {                  // *Регистрация сервиса в контейнере зависимостей как Singleton*                  services.AddSingleton<INotificationService, NotificationService>();                  // *Регистрация основного приложения*                  services.AddSingleton<Application>();              }).Build();              // *Получение экземпляра основного приложения из контейнера зависимостей*              var app = host.Services.GetRequiredService<Application>();              // *Запуск основного приложения*              app.Run();          }      }      public interface INotificationService      {          string GetMessage();      }      public class NotificationService : INotificationService      {          public string GetMessage()          {              return "Some Message";          }      }      public class Application      {          // *Приватное поле для хранения зависимости*          private readonly INotificationService \_notificationService;          // *Конструктор с внедрением зависимости*          public Application(INotificationService notificationService)          {              \_notificationService = notificationService;          }          public void Run()          {              // *Использование внедренного сервиса для получения и вывода сообщения*              Console.WriteLine(\_notificationService.GetMessage());          }      }  } |

**Ссылка на репозиторий:** <https://github.com/avoryyy/backend_mospoly>