**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 №3**

**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:**

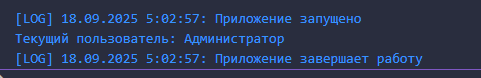
Программа основана на использовании двух сервисов: службы для работы с пользователями и службы для логирования. Интерфейсы IUserService и ILoggerService определяют контракты для этих служб, а классы UserService и ConsoleLoggerService реализуют их. Такой подход позволяет отделить абстракции от реализации и в дальнейшем при необходимости заменять конкретные классы без изменения остальной части программы.

В классе UserService реализован метод GetUserName, который возвращает имя пользователя по идентификатору. Если идентификатор равен единице, выводится «Администратор», иначе используется значение «Пользователь». Класс ConsoleLoggerService отвечает за логирование и выводит сообщения в консоль с указанием времени и префиксом [LOG], что позволяет наглядно отслеживать процесс выполнения программы.

Основной класс Application получает зависимости через конструктор: объекты IUserService и ILoggerService передаются ему при создании. Это позволяет не создавать сервисы внутри самого класса, а использовать уже готовые экземпляры, что соответствует принципу инверсии зависимостей. В методе Run выполняется логирование запуска приложения, затем через IUserService определяется имя пользователя с заданным идентификатором и выводится в консоль, после чего фиксируется завершение работы программы.

В методе Main создаются экземпляры сервисов UserService и ConsoleLoggerService, которые затем передаются в Application.

**Скриншоты работы приложения:**



**Листинг:**

**Program.cs:**

|  |
| --- |
| // *Сервис для работы с пользователями*  public interface IUserService  {      string GetUserName(int userId);  }  public class UserService : IUserService  {      public string GetUserName(int userId)      {          return userId == 1 ? "Администратор" : "Пользователь";      }  }  // *Сервис для логирования*  public interface ILoggerService  {      void Log(string message);  }  public class ConsoleLoggerService : ILoggerService  {      public void Log(string message)      {          Console.WriteLine($"[LOG] {DateTime.Now}: {message}");      }  }  // *Основной класс приложения*  public class Application  {      private readonly IUserService \_userService;      private readonly ILoggerService \_logger;      public Application(IUserService userService, ILoggerService logger)      {          \_userService = userService;          \_logger = logger;      }      public void Run()      {          \_logger.Log("Приложение запущено");          var userName = \_userService.GetUserName(1);          Console.WriteLine($"Текущий пользователь: {userName}");          \_logger.Log("Приложение завершает работу");      }  }  class Program  {      static void Main()      {          // *Создаем экземпляры сервисов*          ILoggerService logger = new ConsoleLoggerService();          IUserService userService = new UserService();          // *Внедряем зависимости через конструктор*          var app = new Application(userService, logger);          app.Run();      }  } |

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