Министерство Образования, Культуры и Исследований Республики Молдова

Технический Университет Молдовы

Факультет Вычислительной Техники, Информатики и Микроэлектроники

Департамент Программной Инженерии и Автоматики

**Лабораторная Работа №1**

по Сетевому Программированию

Тема: «**Изучение технологии контейнеризации. Введение в Docker.»**

Выполнил Кара Наталья

ст. гр. SI-222

Проверил

Макеев Никита

Кишинев – 2024

## Цель работы.

Изучить технологии контейнеризации, введение в Docker.

## Теоретическая часть.

1. Понятие контейнеризации.

Контейнеризация — это технология, которая позволяет изолировать приложение и все его зависимости (библиотеки, настройки, переменные окружения) в легковесной, переносимой и самодостаточной среде, называемой контейнером. Контейнеры работают на уровне операционной системы и используют общее ядро ОС, что делает их более эффективными по сравнению с виртуальными машинами.

Контейнеры обеспечивают:

1. Изоляцию: Каждое приложение работает в своей среде, не влияя на другие приложения или хост-систему.
2. Переносимость: Контейнеры могут быть запущены на любой системе, поддерживающей Docker или другую платформу контейнеризации.
3. Повторяемость: Контейнеры гарантируют, что приложение будет работать одинаково в любой среде (разработка, тестирование, production).

2. Преимущества по сравнению с виртуальными машинами.

1. Легковесность:

Контейнеры занимают меньше места и быстрее запускаются, так как используют общее ядро ОС.

1. Переносимость:

Контейнеры работают одинаково на любой системе, поддерживающей Docker (разработка, тестирование, production).

1. Изоляция:

Каждое приложение работает в своей среде, что предотвращает конфликты зависимостей.

1. Масштабируемость:

Контейнеры легко масштабируются с помощью оркестраторов, таких как Kubernetes.

1. Эффективность ресурсов:

Контейнеры потребляют меньше ресурсов по сравнению с виртуальными машинами.

1. Упрощение CI/CD:

Контейнеры упрощают процесс непрерывной интеграции и доставки (CI/CD), так как обеспечивают одинаковую среду на всех этапах.

1. Основные команды Docker

Dockerfile: Файл с инструкциями для создания Docker-образа.

1. docker build: Сборка Docker-образа на основе Dockerfile.
2. docker run: Запуск контейнера из образа.
3. docker ps: Просмотр запущенных контейнеров.
4. docker images: Просмотр списка образов.
5. docker stop: Остановка контейнера.
6. docker rm: Удаление контейнера.
7. docker rmi: Удаление образа.

## Ход работы.

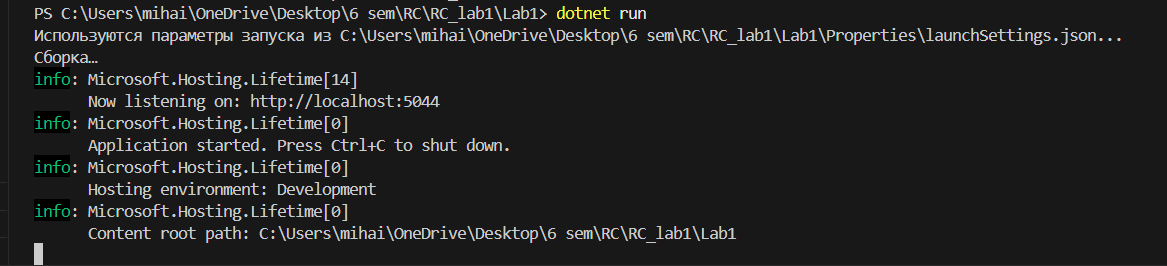
1. Создание простого веб-приложения.

Для создания приложения был выбран ASP.Net Core.

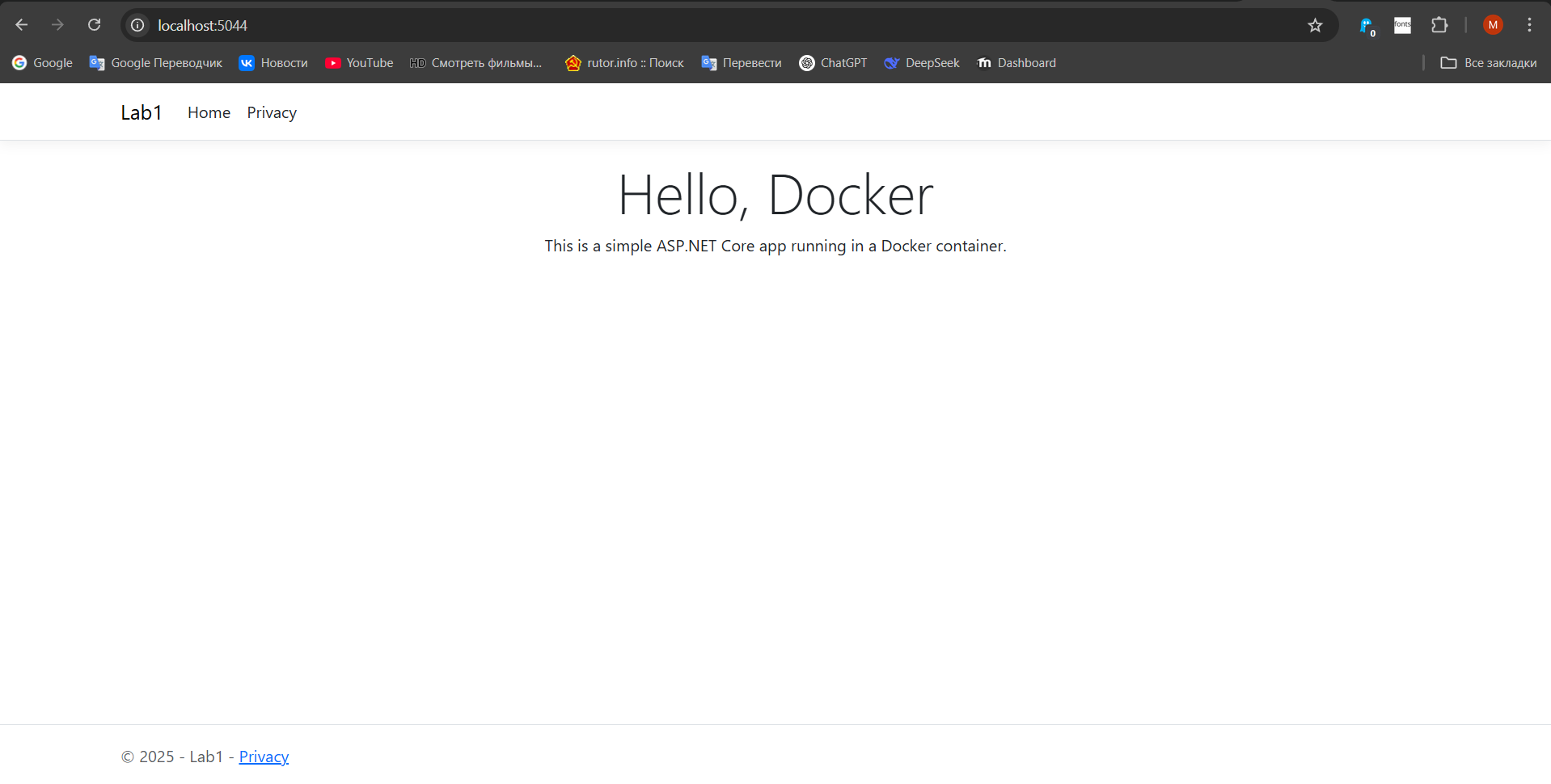
Простое веб-приложение создаётся с помощью команды:

dotnet new webapp -o Lab1.

Для проверки вводится команда: dotnet run

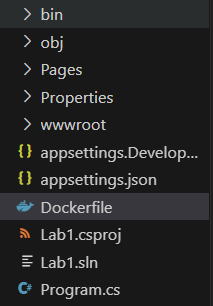


В браузере открывается ссылка, в моем случае: <http://localhost:5044/>

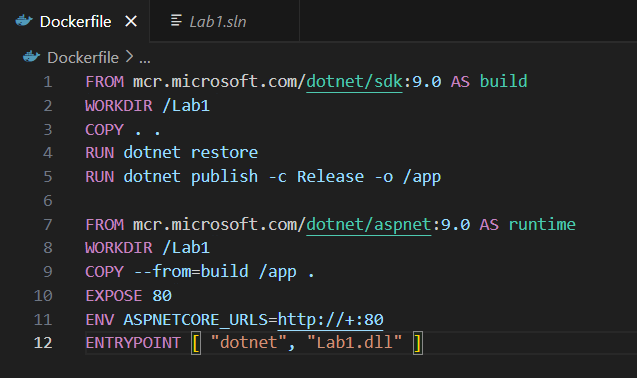


1. Написание Dockerfile.

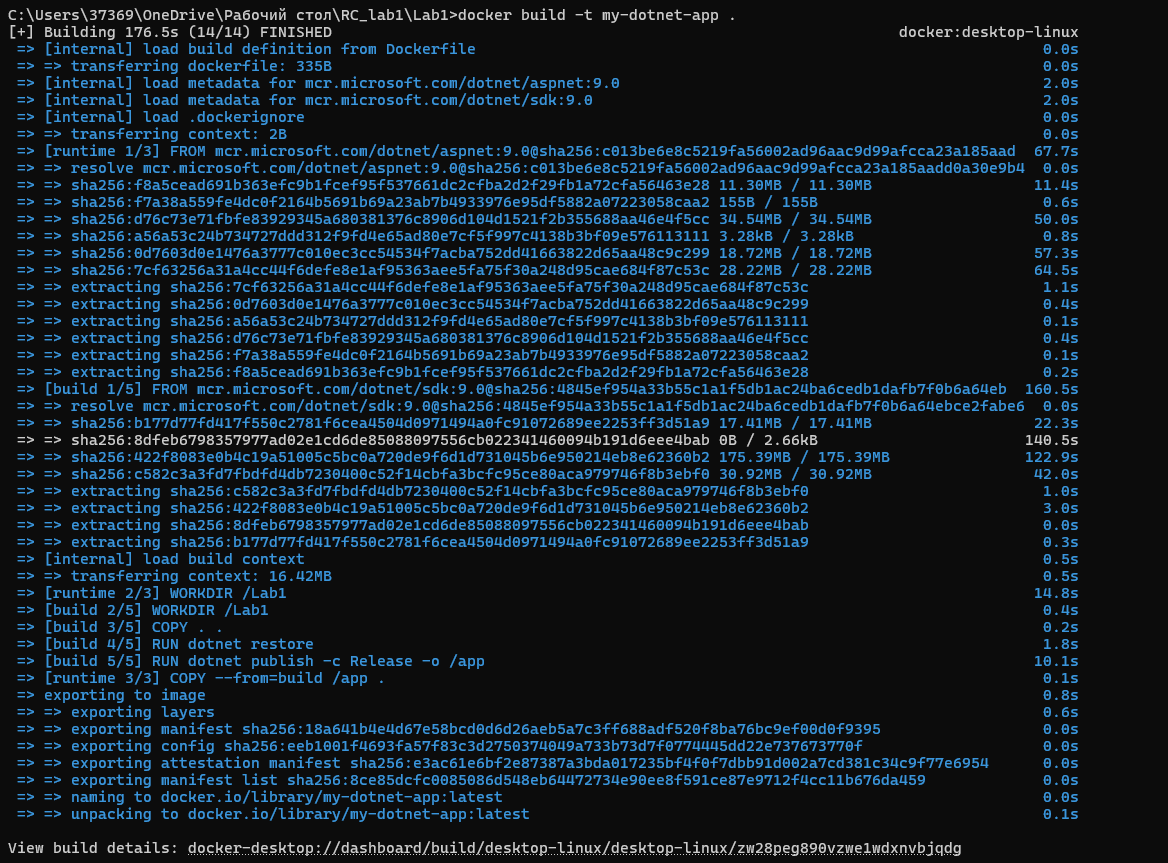
Для создания Dockerfile создаётся новый файл в корневой папке проекта под названием Dockerfile.

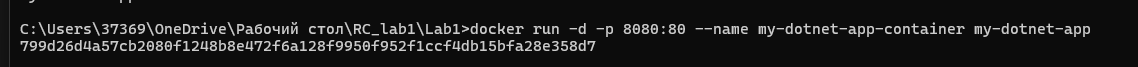


Написание Dockerfile

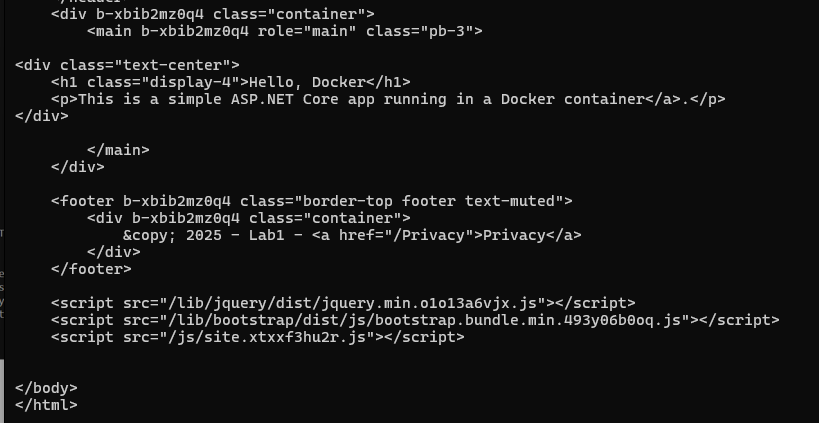


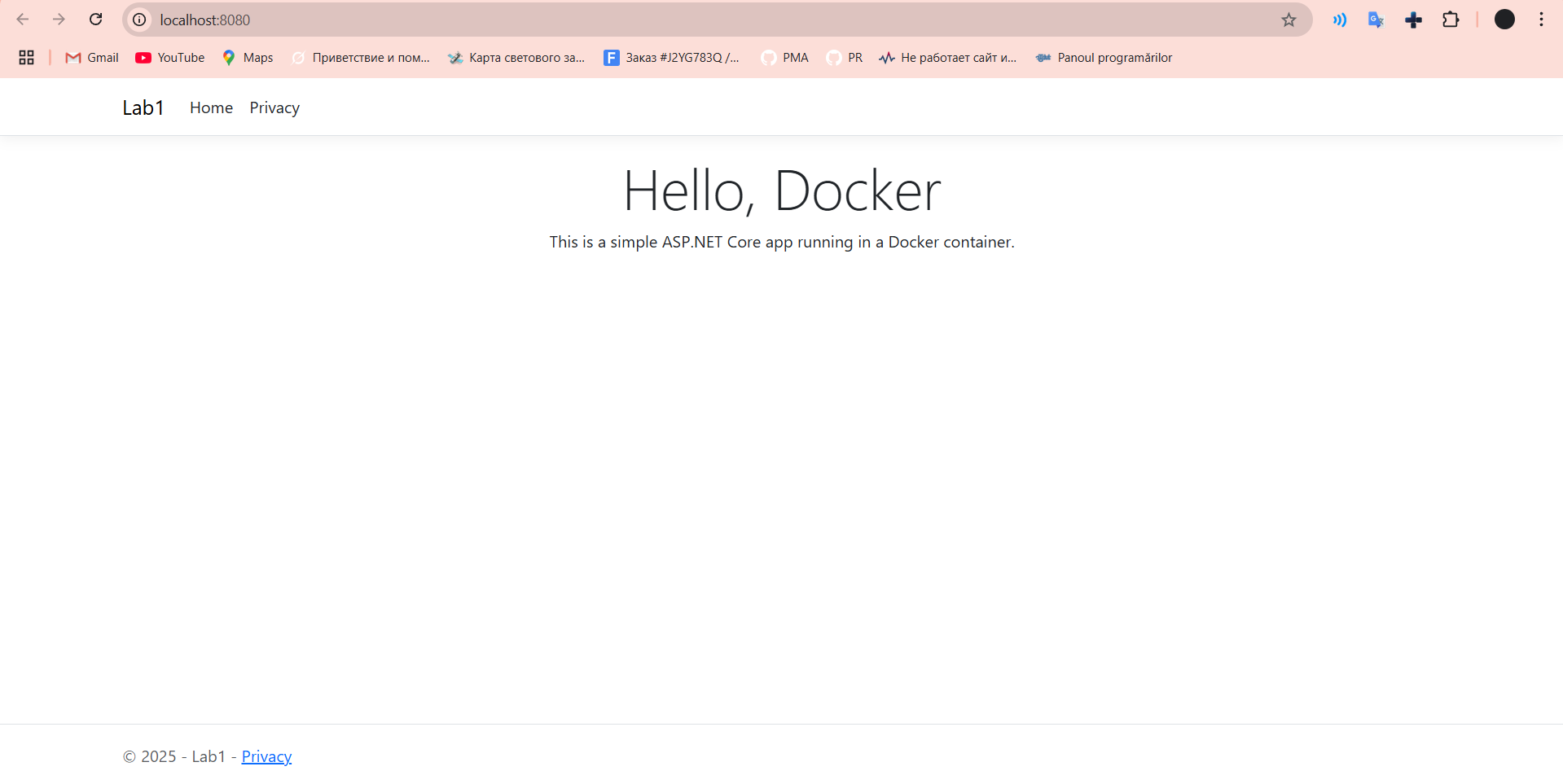
1. Сборка образа и запуск контейнера.





1. Проверка с помощью curl.  
   

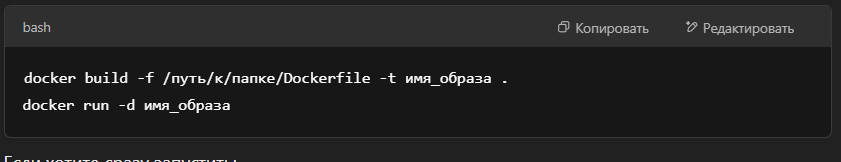




## Выводы.

В ходе выполнения лабораторной работы я разработал простое веб-приложение на ASP.NET Core и успешно запустил его в Docker-контейнере. В процессе работы я изучил основы технологии контейнеризации, включая создание Dockerfile, сборку образов и запуск контейнеров. Я убедился, что контейнеризация позволяет легко развертывать приложения в изолированной среде, что значительно упрощает процесс разработки, тестирования и переноса приложений между различными средами.

Кроме того, я познакомился с преимуществами контейнеризации по сравнению с виртуальными машинами, такими как легковесность, быстрый запуск и эффективное использование ресурсов. Этот опыт позволил мне лучше понять, как современные технологии упрощают разработку и deployment приложений, а также как Docker может быть полезен в реальных проектах.  
  
Запустить образ из другой папки



Вывод всех контейнеров docker ps -a

## Приложение.

<https://github.com/NatashkaJK/PR1>