
Архитектура распределённых приложений

Лабораторная работа №1

Выполнил: студент группы ИП-312

Дорогин Н. С.

Проверил: преподаватель Кафедры вычислительных систем

Челканова Т. В.

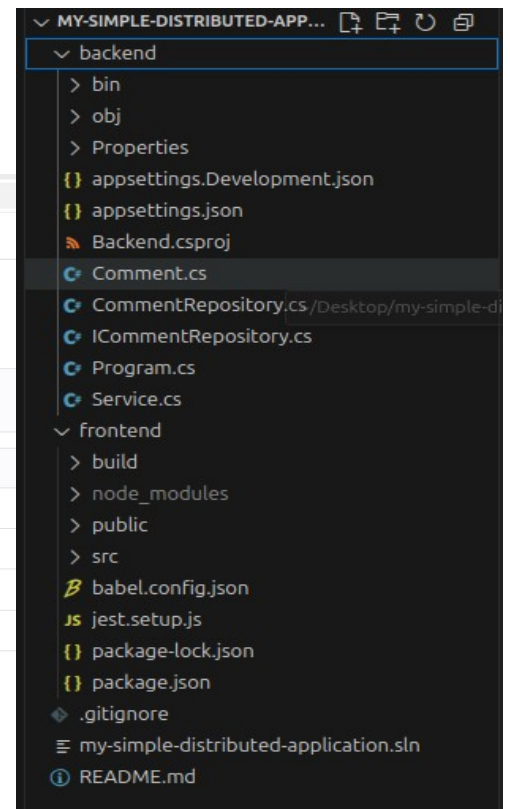
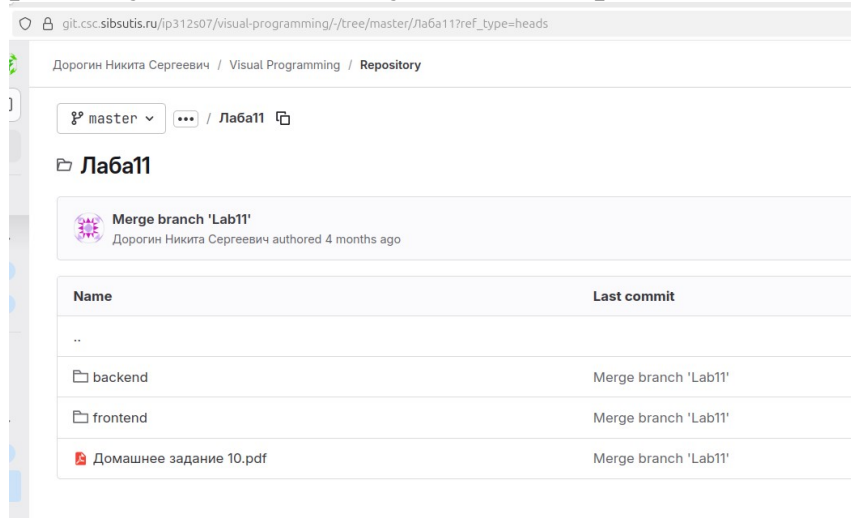
Задание:

1. Получить доступ к Gitlab кафедры вычислительных систем.
 2. Разработать приложение, реализующее простой API по протоколу http. Допускается взять уже существующее.
 3. Приложение должно корректно обрабатывать сигналы остановки.
 4. Написать для приложения Dockerfile для сборки образа контейнера.
 5. Продемонстрировать работу приложения в контейнере. Для этого необходимо запустить контейнер и обеспечить доступ по сети.
-

Ход работы:

1, 2, 3) Уже имея доступ к Gitlab кафедры вычислительных систем с прошлых курсов, создаю там репозитой my-simple-distributed-application в котором и будет вестись работа в рамках этого курса.

В качестве реализации API по протоколу http возьму приложение с прошлого курса, реализующее таблицу комментариев.



Устанавливаем node.js для запуска React-приложения, реализующего фронтенд и ASP.NET Core для запуска бэкэнда (сервера).

```
gastello123@gastello123:~/Desktop/my-simple-distributed-application/frontend$ npm install react@latest react-dom@latest
npm WARN ERESOLVE overriding peer dependency
npm WARN While resolving: undefined@undefined
npm WARN Found: react@0.0.0-experimental-6a7650c7-20250405
npm WARN node_modules/react
npm WARN   peer react@"0.0.0-experimental-6a7650c7-20250405" from react-dom@0.0.0-experimental-6a7650c7-20250405
npm WARN   node_modules/react-dom
npm WARN     react-dom@"19.1.1" from the root project
npm WARN 1 more (the root project)
npm WARN Could not resolve dependency:
npm WARN peer react@"^18.0.0" from @testing-library/react@14.3.1
npm WARN node_modules/@testing-library/react
npm WARN   dev @testing-library/react@"^14.0.0" from the root project
npm WARN ERESOLVE overriding peer dependency
npm WARN While resolving: undefined@undefined
npm WARN Found: react-dom@0.0.0-experimental-6a7650c7-20250405
npm WARN node_modules/react-dom
npm WARN   react-dom@"19.1.1" from the root project
npm WARN Could not resolve dependency:
npm WARN peer react-dom@"^18.0.0" from @testing-library/react@14.3.1
```

```
Run npm audit for details.
gastello123@gastello123:~/Desktop/my-simple-distributed-application/frontend$ npm install react-scripts
up to date, audited 1486 packages in 3s

273 packages are looking for funding
  run `npm fund` for details

14 vulnerabilities (3 low, 4 moderate, 6 high, 1 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
gastello123@gastello123:~/Desktop/my-simple-distributed-application/frontend$
```

```

gastello123@gastello123:~/Desktop$ wget https://dot.net/v1/dotnet-install.sh -O dotnet-install.sh && chmod +x dotnet-install.sh && ./dotnet-install.sh --channel 9.0 && echo 'export DOTNET_ROOT=$HOME/.dotnet' >> ~/.bashrc && echo 'export PATH=$PATH:$HOME/.dotnet:$HOME/.dotnet/tools' >> ~/.bashrc && source ~/.bashrc

--2025-09-20 21:57:46-- https://dot.net/v1/dotnet-install.sh
Распознаётся dot.net (dot.net)... 20.236.44.162, 20.231.239.246, 20.70.246.20, ...
Подключение к dot.net (dot.net)[20.236.44.162]:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 301 Moved Permanently
Адрес: https://builds.dotnet.microsoft.com/dotnet/scripts/v1/dotnet-install.sh [переход]
--2025-09-20 21:57:47-- https://builds.dotnet.microsoft.com/dotnet/scripts/v1/dotnet-install.sh
Распознаётся builds.dotnet.microsoft.com (builds.dotnet.microsoft.com)... 23.196.236.91, 23.196.236.82, 2001:2000:0:17::50ef:8a92, ...
Подключение к builds.dotnet.microsoft.com (builds.dotnet.microsoft.com)[23.196.236.91]:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [application/octet-stream]
Сохранение в: 'dotnet-install.sh'

dotnet-install.sh [ <=> ] 62.15K ...KB/s за
0.03s

2025-09-20 21:57:48 (2.14 MB/s) - 'dotnet-install.sh' сохранён [63644]

dotnet-install: Attempting to download using aka.ms link https://builds.dotnet.microsoft.com/dotnet/Sdk/9.0.305/dotnet-sdk-9.0.305-linux-x64.tar.gz
dotnet-install: Remote file https://builds.dotnet.microsoft.com/dotnet/Sdk/9.0.305/dotnet-sdk-9.0.305-linux-x64.tar.gz size is 218127088 bytes.
gastello123@gastello123:~/Desktop/my-simple-distributed-application/backend$ export DOTNET_ROOT=$HOME/.dotnet && export PATH=$HOME/.dotnet:$HOME/.dotnet/tools:$PATH

gastello123@gastello123:~/Desktop/my-simple-distributed-application/backend$ dotnet --version
9.0.305
gastello123@gastello123:~/Desktop/my-simple-distributed-application/backend$ echo 'export DOTNET_ROOT=$HOME/.dotnet' >> ~/.bashrc && echo 'export PATH=$HOME/.dotnet:$HOME/.dotnet/tools:$PATH' >> ~/.bashrc && source ~/.bashrc

gastello123@gastello123:~/Desktop/my-simple-distributed-application/backend$ dotnet run

Вас приветствует .NET 9.0!

```

Запускаем приложение:

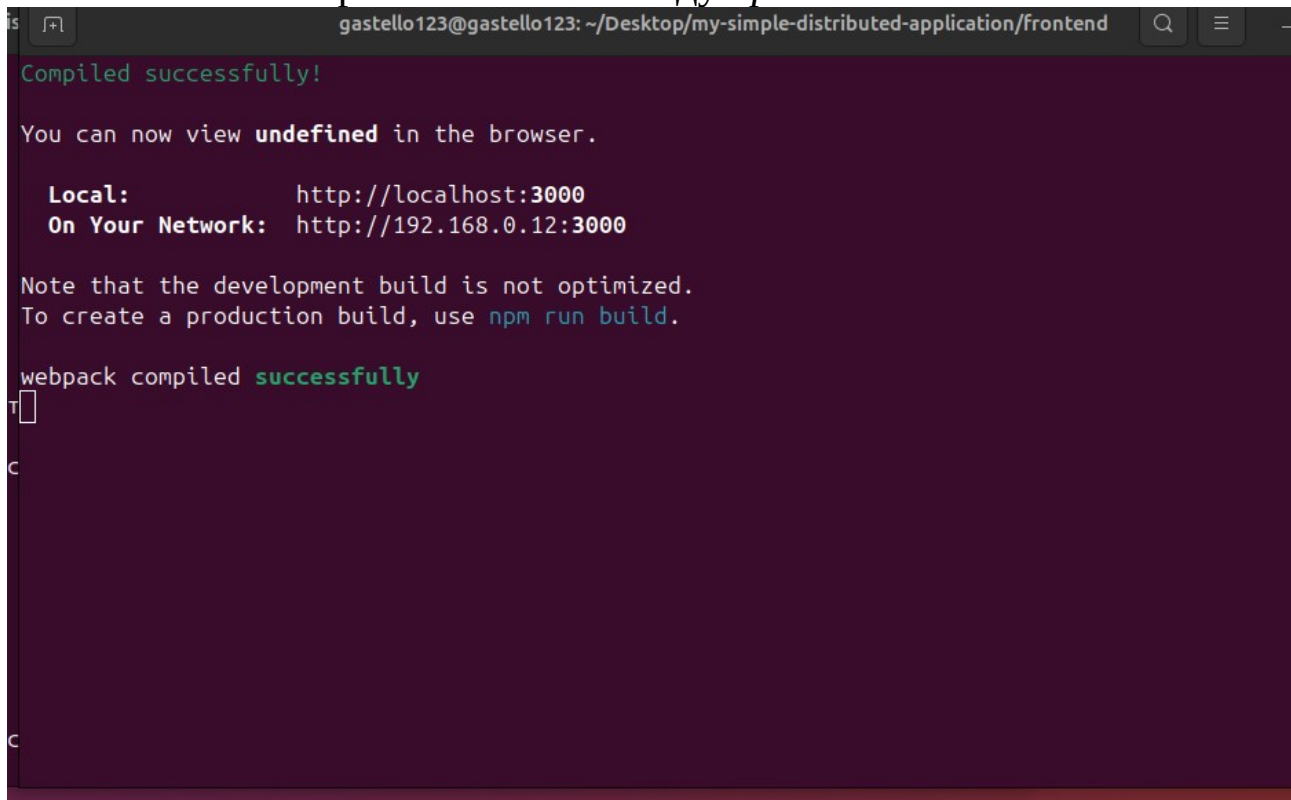
В папке backend прописываем команду *dotnet run*

```

gastello123@gastello123: ~/Desktop/my-simple-dis
-----
Установлен сертификат разработки ASP.NET Core HTTPS.
Чтобы доверять сертификату, выполните команду "dotnet dev-certs https --trust"
Дополнительные сведения об HTTPS: https://aka.ms/dotnet-https
-----
Как написать свое первое приложение: https://aka.ms/dotnet-hello-world
Узнать о новых возможностях: https://aka.ms/dotnet-whats-new
Просмотреть документацию: https://aka.ms/dotnet-docs
Сообщить о проблемах и найти исходный код на GitHub: https://github.com/dotnet/core
Для просмотра доступных команд введите команду "dotnet --help" или посетите следующую ст
-----
Используются параметры запуска из /home/gastello123/Desktop/my-simple-distributed-applic
Сборка...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5139
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /home/gastello123/Desktop/my-simple-distributed-application/bac

```

В папке frontend прописываем команду *npm start*



```
gastello123@gastello123: ~/Desktop/my-simple-distributed-application/frontend
Compiled successfully!

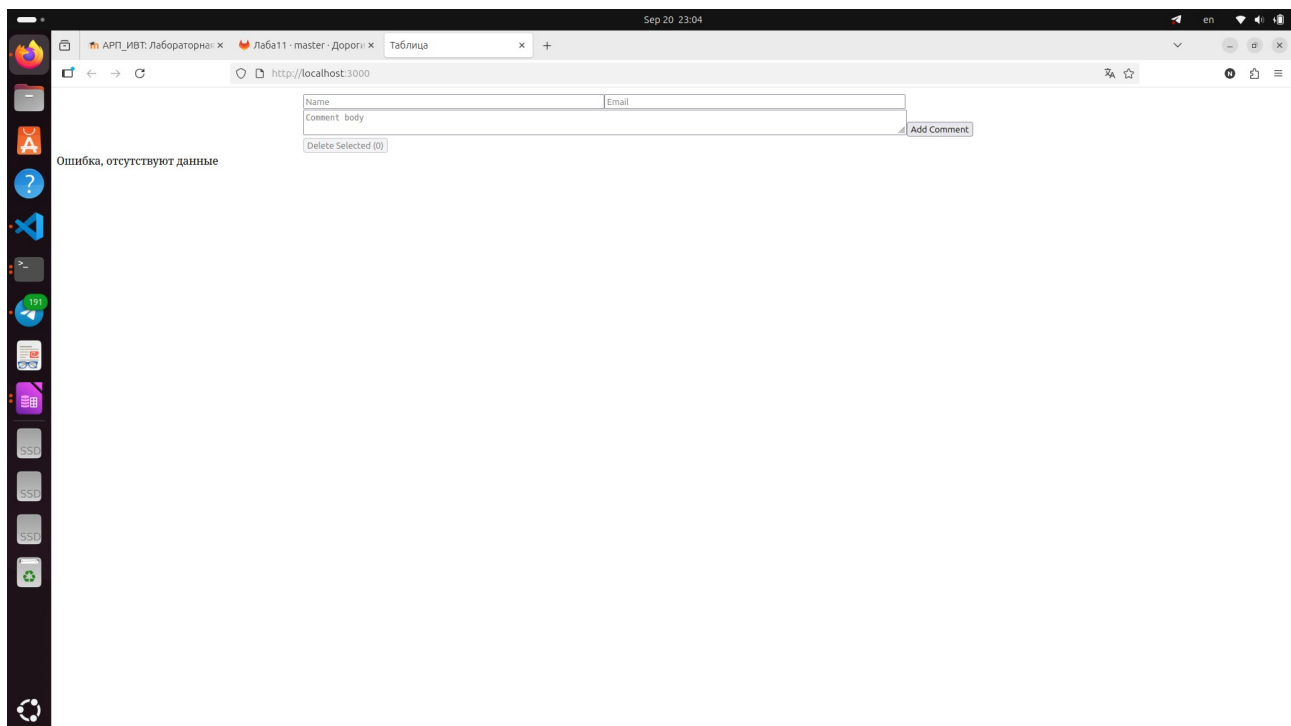
You can now view undefined in the browser.

Local:      http://localhost:3000
On Your Network:  http://192.168.0.12:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Приложение открывается:



Пока что в таблицу ничего не занесено, поэтому выводится сообщение об отсутствии данных.

Никита

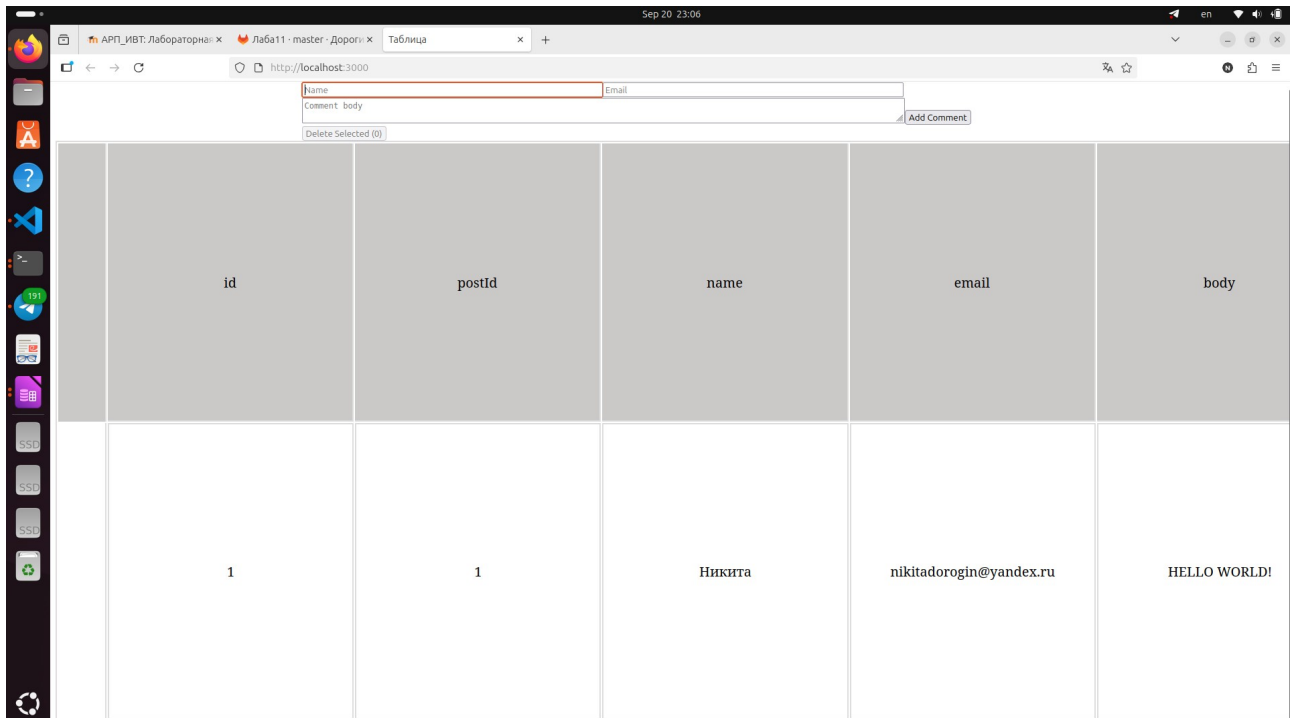
nikitadorogin@yandex.ru

HELLO WORLD!

Delete Selected (0)

Add Comment

Вписав корректные данные и нажав AddComment мы отправим на сервер http запрос POST и таким образом добавим туда наши данные.

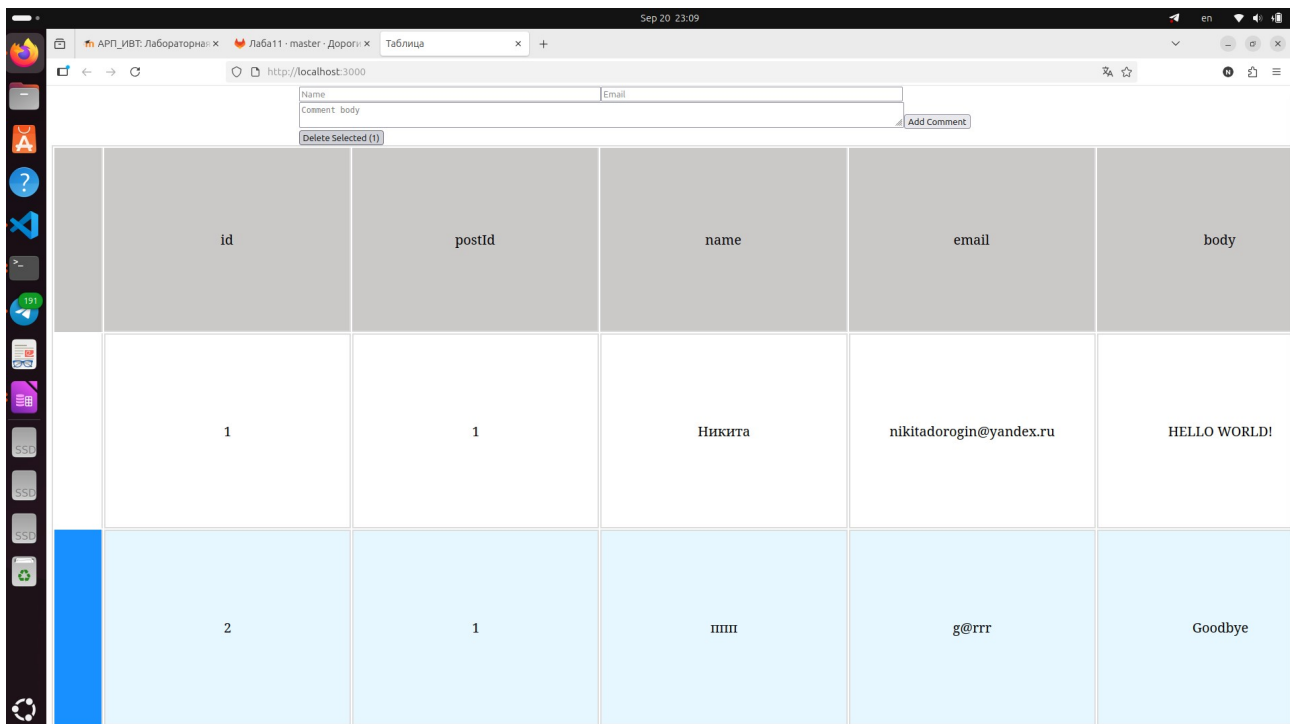


The screenshot shows a web browser window with a table. The table has five columns: id, postId, name, email, and body. The first row contains the values 1, 1, Никита, nikitadorogin@yandex.ru, and HELLO WORLD!. Above the table is a form with fields for Name, Email, and Comment, and an Add Comment button.

id	postId	name	email	body
1	1	Никита	nikitadorogin@yandex.ru	HELLO WORLD!

Нажав несколько раз на ячейку и изменив в ней значение мы отправим на сервер http запрос PATCH, изменив уже имеющиеся там данные.

1	1	Никита	nikitadorogin@yandex.ru	HELLO WORLD!
---	---	--------	-------------------------	--------------



Выбрав строку и нажав кнопку Delete Selected мы отправим на сервер http запрос DELETE и таким образом удалим оттуда выбранные данные.

При этом на протяжении всей работы приложения React-приложение посылает серверу http запрос GET, чтобы получать данные, которые оно выводит в таблице.

4) Dockerfile для приложения выглядит следующим образом:

```
#my-dockerfile
```

```
# Указываем систему, с которой запускаем  
FROM ubuntu:24.04
```

```
# Установка системных зависимостей  
RUN apt-get update  
RUN apt-get install -y wget curl gnupg software-  
properties-common  
RUN rm -rf /var/lib/apt/lists/*
```

```
# Установка React
RUN apt-get update
RUN apt-get install -y nodejs npm

# Установка ASP.NET Core
RUN curl -sSL https://dot.net/v1/dotnet-install.sh
| bash /dev/stdin --channel 9.0 --install-dir
/usr/share/dotnet
# Добавляем .NET в PATH
ENV DOTNET_ROOT=/usr/share/dotnet
ENV PATH=$PATH:/usr/share/dotnet

# Создаём рабочую директорию и переходим в неё
WORKDIR /app

# Копируем из исходников в рабочую директорию
package.json из фронтенда
COPY src/frontend/package*.json ./src/frontend/
# и основной файл C# проекта из бэкенда
COPY src/backend/Backend.csproj ./src/backend/

# Восстановление зависимостей бэкенда
RUN dotnet restore ./src/backend/Backend.csproj

# Устанавливаем зависимости для Реакта.
RUN cd ./src/frontend && npm ci

# Копируем код из исходника в рабочую директорию
COPY . .

# Компилируем и собираем бэкенд (в папку
./publish)
RUN dotnet publish ./src/backend/Backend.csproj -c
Release -o ./publish

# Собираем бэкенд
```



```
RUN cd src/frontend && npm run build
```

```
# Создаём в бэкенде если нет папку wwwroot для  
хранения статических файлов фронтенда
```

```
RUN mkdir -p ./publish/wwwroot/ && \  
    cp -r src/frontend/build/* ./publish/wwwroot/
```

```
# Перемещаемся в папку с собранным бэкендом  
WORKDIR /app/publish
```

```
# Ставим порт, через который будем получать доступ  
к приложению.
```

```
ENV ASPNETCORE_URLS=http://0.0.0.0:5000
```

```
EXPOSE 5000
```

```
# Устанавливаем точку входа.
```

```
ENTRYPOINT ["dotnet", "Backend.dll"]
```

5) Проверяем работу приложения в контейнере.

Собираем образ нашего приложения

```
gastello123@gastello123:~/Desktop/my-simple-distributed-application$ sudo docker build -t my-simple-distributed-application .  
[+] Building 2.7s (22/22) FINISHED  
=> [internal] load build definition from Dockerfile  
=> => transferring dockerfile: 2.20kB  
=> [internal] load metadata for docker.io/library/ubuntu:24.04  
=> [internal] load .dockerignore  
=> => transferring context: 2B  
=> [ 1/17] FROM docker.io/library/ubuntu:24.04@sha256:353675e2a41babd526e2b837d7ec780c2a05bca0164f7ea5dbbd433d21d166fc  
=> [internal] load build context  
=> => transferring context: 3.98MB  
=> CACHED [ 2/17] RUN apt-get update  
=> CACHED [ 3/17] RUN apt-get install -y wget curl gnupg software-properties-common  
=> CACHED [ 4/17] RUN rm -rf /var/lib/apt/lists/*  
=> CACHED [ 5/17] RUN apt-get update  
=> CACHED [ 6/17] RUN apt-get install -y nodejs npm  
=> CACHED [ 7/17] RUN curl -sSL https://dot.net/v1/dotnet-install.sh | bash /dev/stdin --channel 9.0 --install-dir /usr/share/dotnet  
=> CACHED [ 8/17] WORKDIR /app  
=> CACHED [ 9/17] COPY src/frontend/package*.json ./src/frontend/  
=> CACHED [10/17] COPY src/backend/Backend.csproj ./src/backend/  
=> CACHED [11/17] RUN dotnet restore ./src/backend/Backend.csproj  
=> CACHED [12/17] RUN cd ./src/frontend && npm ci  
=> CACHED [13/17] COPY . .  
=> CACHED [14/17] RUN dotnet publish ./src/backend/Backend.csproj -c Release -o ./publish  
=> CACHED [15/17] RUN cd src/frontend && npm run build  
=> CACHED [16/17] RUN mkdir -p ./publish/wwwroot/ && cp -r src/frontend/build/* ./publish/wwwroot/  
=> CACHED [17/17] WORKDIR /app/publish  
=> exporting to image  
=> => exporting layers  
=> => writing image sha256:8124e7afaff9b427c9d442f31c1490b9e49336eaf82107a7fb9683e58f67720b  
=> => naming to docker.io/library/my-simple-distributed-application  
gastello123@gastello123:~/Desktop/my-simple-distributed-application$
```

(сборка проходит быстро, очевидно ввиду того, что кэш помнит прошлые попытки)

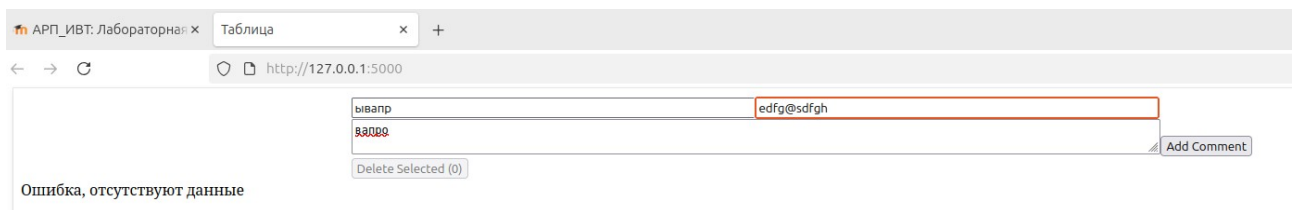
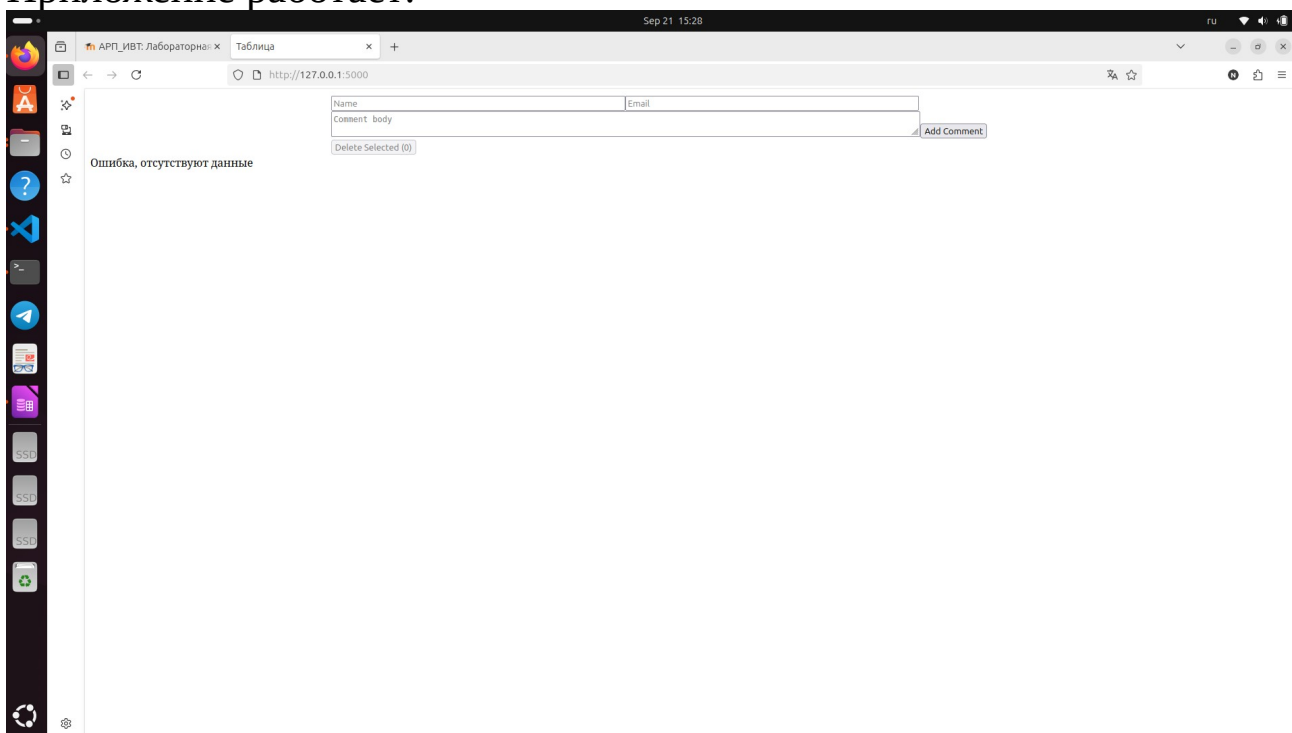
Создаём контейнер:

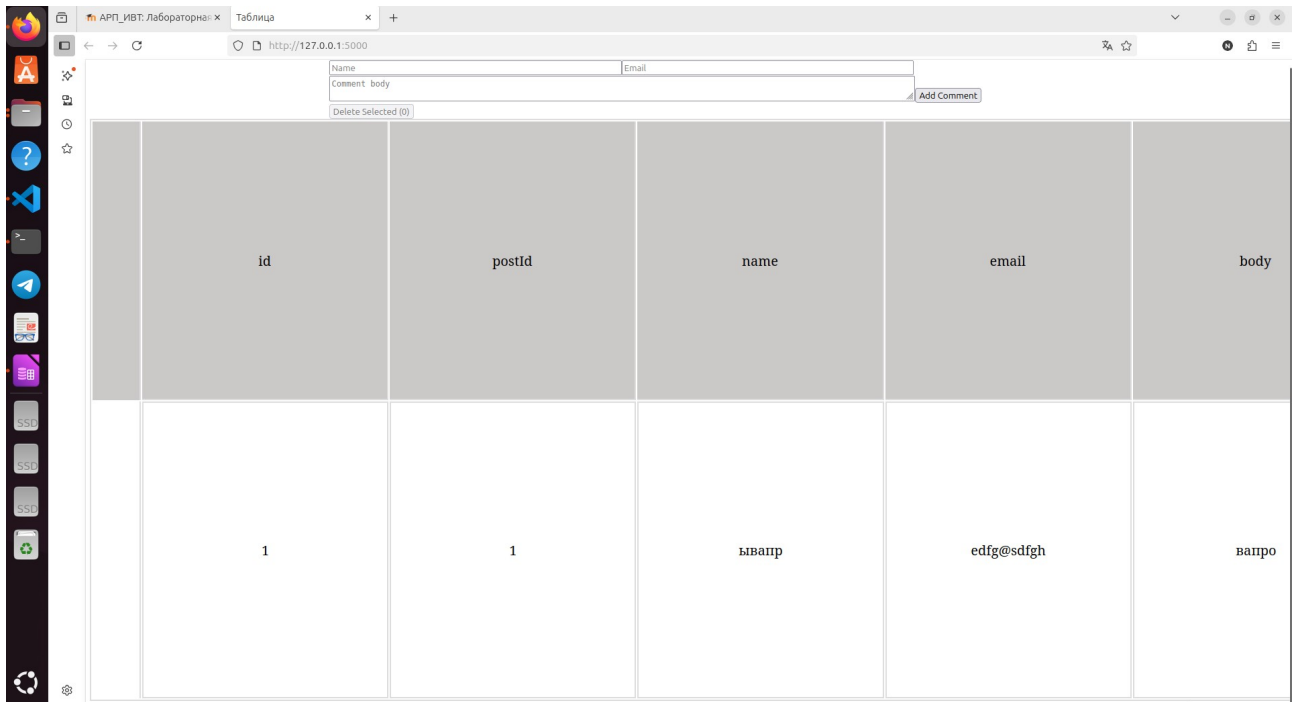
```
gastello123@gastello123: ~/Desktop/my-simple-distributed-application$ sudo docker create my-simple-distributed-application  
a62598d32a4cc2ae7644a9d63eebc2390508dd72539be3736c24047e2bdef81f
```

Запускаем контейнер:

```
gastello123@gastello123:~/Desktop/my-simple-distributed-application$ sudo docker run -p 5000:5000 my-simple-distributed-application  
info: Microsoft.Hosting.Lifetime[14]  
Now listening on: http://0.0.0.0:5000  
info: Microsoft.Hosting.Lifetime[0]  
Application started. Press Ctrl+C to shut down.  
info: Microsoft.Hosting.Lifetime[0]  
Hosting environment: Production  
info: Microsoft.Hosting.Lifetime[0]  
Content root path: /app/publish
```

Приложение работает:





Теперь зайдя с другого терминала попробуем остановить контейнер:

```
gastello123@gastello123: ~/Desktop/my-simple-distributed-application
gastello123@gastello123: $ cd ~/Desktop/my-simple-distributed-application
gastello123@gastello123:~/Desktop/my-simple-distributed-application$ sudo docker ps
[sudo] пароль для gastello123:
CONTAINER ID   IMAGE                                COMMAND                  CREATED
STATUS        PORTS
14ace460cdfd   my-simple-distributed-application   "dotnet Backend.dll"    20 seconds ago
Up 19 seconds  0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   blissful_brattain
gastello123@gastello123:~/Desktop/my-simple-distributed-application$ sudo docker stop blissful_brattain
gastello123@gastello123:~/Desktop/my-simple-distributed-application$

gastello123@gastello123: ~/Desktop/my-simple-distributed-application
gastello123@gastello123:~/Desktop/my-simple-distributed-application$ sudo docker run -p 5000 my-simple-distributed-application
info: Microsoft.Hosting.Lifetime[14]
0:5000 my-simple-distributed-application
Now listening on: http://0.0.0.0:5000
info: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
Content root path: /app/publish
info: Microsoft.Hosting.Lifetime[0]
Application is shutting down...
gastello123@gastello123:~/Desktop/my-simple-distributed-application$
```

Приложение сразу же закроется и во фронте, попробовав обновить страницу или что-то ещё получим

