

Отчет по большому домашнему заданию №2 по курсу «Конструирование программного обеспечения»

Каспари Давид Эрвинович.

В этом отчете я отражаю результаты своей работы, опишу архитектуру написанной программы, приведу инструкции.

1. Требования к функционалу.

3-й пункт необязателен, а второй в процессе разбирательств тоже стал необязательным (С. А. Виденин подтвердил в беседе). Поэтому мной был реализован только первый пункт по подсчёту статистики.

2. Микросервисная архитектура.

Было реализовано 2 микросервиса из предложенных, так как FileStorage не нужен без проверки файлов на схожесть.

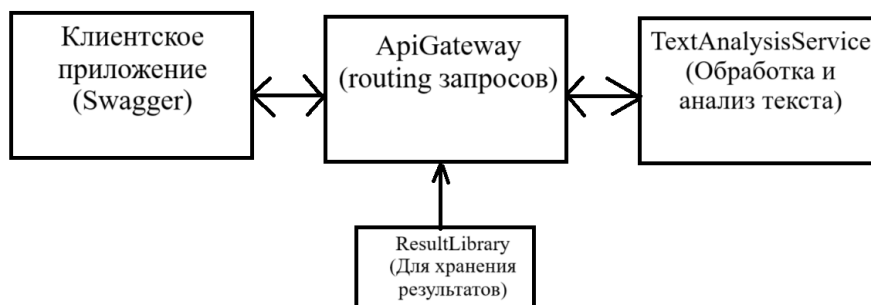
Реализованные сервисы:

- 1) API Gateway – отвечает только за routing запросов
- 2) Text Analysis Service – отвечает только за проведение анализа, хранение результатов и их выдачу

3. Архитектура программы.

В Visual Studio созданы Веб-API Asp.Net.Core с названиями ApiGateway и TextAnalysisService. Также создана библиотека классов ResultLibrary.

На следующем рисунке показана общая схема работы:



Библиотека классов ResultLibrary содержит класс AnalysisResult, который отвечает за представление результатов анализа данных.

ApiGateway содержит Program.cs, который занимается маршрутизацией.

TextAnalysisService содержит Program.cs, занимающийся обработкой запроса, в Services он содержит

ITextAnalysisService.cs – интерфейс для класса, который занимается анализом, и TextAnalysisService.cs, который содержит методы для подсчета абзацев, слов, символов.

Более подробно с кодом можно ознакомиться и с помощью комментариев в программах.

4. Сборка и запуск программы. Docker-файлы.

Программа собиралась через Docker. Было написано два Dockerfile для ApiGateway и TextAnalysisService, а также docker-compose.yml, собирающий программу.

Они были написаны с помощью инструкции, прикрепленной к материалам к этому домашнему заданию.

Приведу скриншоты процесса сборки.

После docker-compose build:

```
=> [text-analysis] resolving provenance for metadata file 0.0s
=> [api-gateway internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 648B 0.0s
=> [api-gateway internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [api-gateway internal] load build context 0.0s
=> => transferring context: 11.38kB 0.0s
=> CACHED [api-gateway build 3/8] COPY [ResultLibrary/ResultLibrary.csproj, ResultLibrary/] 0.0s
=> CACHED [api-gateway build 4/8] COPY [ApiGateway/ApiGateway.csproj, ApiGateway/] 0.0s
=> CACHED [api-gateway build 5/8] RUN dotnet restore "ApiGateway/ApiGateway.csproj" 0.0s
=> [api-gateway build 6/8] COPY . . 0.1s
=> [api-gateway build 7/8] WORKDIR /src/ApiGateway 0.1s
=> [api-gateway build 8/8] RUN dotnet build "ApiGateway.csproj" -c Release -o /app/build 9.4s
=> [api-gateway publish 1/1] RUN dotnet publish "ApiGateway.csproj" -c Release -o /app/publish 7.9s
=> CACHED [api-gateway stage-2 3/3] COPY --from=publish /app/publish . 0.0s
=> [api-gateway] exporting to image 0.2s
=> => exporting layers 0.0s
=> => exporting manifest sha256:9116620610bbd5c23639bb8c7110cc413e2f4c2588163ab92903e463ca821a59 0.0s
=> => exporting config sha256:9124651d9141e0b7b6128bf848e47dafad1dbd995070244fae89d034a15783a6 0.0s
=> => exporting attestation manifest sha256:35ac23a26f3c566adfbb69dcb68773995fd320e709375270374e3b08fcee66ed 0.1s
=> => exporting manifest list sha256:973c13c5e0793259bd7870971c622331ad540b8a3f30ba3db7cdd51b350d93a3 0.0s
=> => naming to docker.io/library/antiplagiatsystem-api-gateway:latest 0.0s
=> => unpacking to docker.io/library/antiplagiatsystem-api-gateway:latest 0.0s
=> [api-gateway] resolving provenance for metadata file 0.0s
[+] Building 2/2
✓api-gateway Built 0.0s
✓text-analysis Built 0.0s
C:\Users\david\source\repos\AntiPlagiatSystem>
```

Программа собралась.

После docker-compose up:

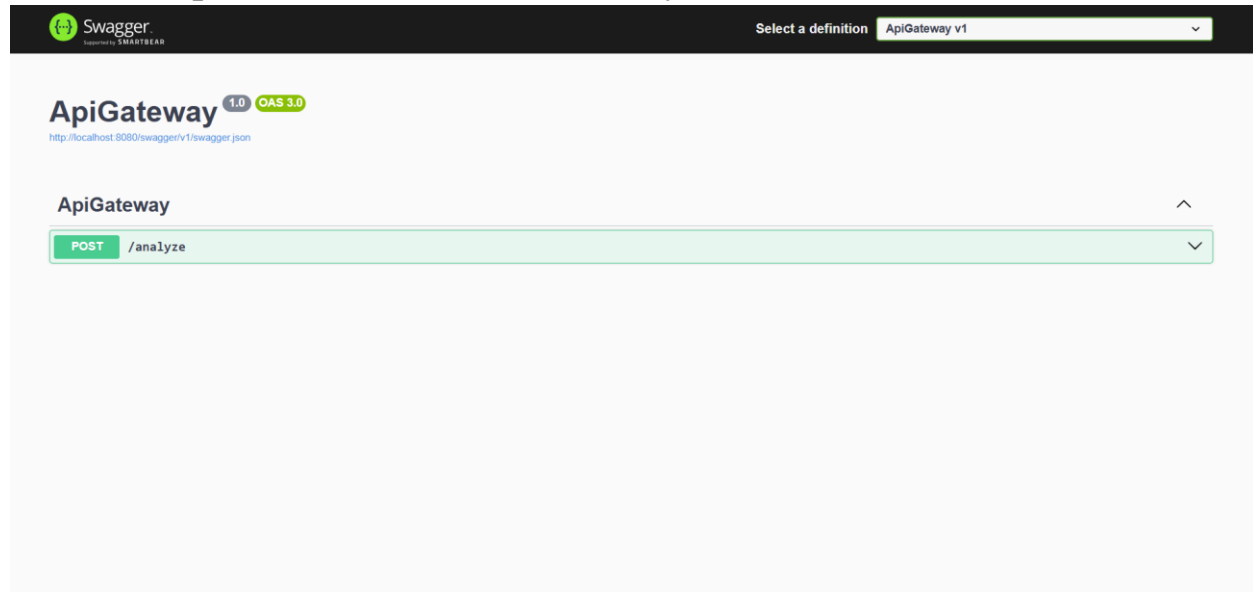
```
✓text-analysis Built 0.0s
C:\Users\david\source\repos\AntiPlagiatSystem>docker-compose up
time="2025-05-22T02:31:37+03:00" level=warning msg="C:\Users\david\source\repos\AntiPlagiatSystem\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 3/3
  ✓ Network antiplagiatsystem_default Created 0.1s
  ✓ Container antiplagiatsystem-text-analysis-1 Created 0.2s
  ✓ Container antiplagiatsystem-api-gateway-1 Created 0.2s
Attaching to api-gateway-1, text-analysis-1
text-analysis-1 | warn: Microsoft.AspNetCore.Hosting.Diagnostics[15]
text-analysis-1 | Overriding HTTP_PORTS '8080' and HTTPS_PORTS ''. Binding to values defined by URLs instead 'http://*:80'.
text-analysis-1 | info: Microsoft.Hosting.Lifetime[14]
text-analysis-1 | Now listening on: http://[::]:80
text-analysis-1 | info: Microsoft.Hosting.Lifetime[0]
text-analysis-1 | Application started. Press Ctrl+C to shut down.
text-analysis-1 | info: Microsoft.Hosting.Lifetime[0]
text-analysis-1 | Hosting environment: Production
text-analysis-1 | info: Microsoft.Hosting.Lifetime[0]
text-analysis-1 | Content root path: /app
api-gateway-1 | warn: Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRepository[60]
api-gateway-1 | Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that may not be persisted outside of the container. Protected data will be unavailable when container is destroyed. For more information go to https://aka.ms/aspnet/dataprotectionwarning
api-gateway-1 | warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
api-gateway-1 | No XML encryptor configured. Key {411d36c7-805a-49e9-a635-614b5a396591} may be persisted to storage in unencrypted form.
api-gateway-1 | warn: Microsoft.AspNetCore.Hosting.Diagnostics[15]
api-gateway-1 | warn: Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRepository[60]
api-gateway-1 | Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that may not be persisted outside of the container. Protected data will be unavailable when container is destroyed. For more information go to https://aka.ms/aspnet/dataprotectionwarning
api-gateway-1 | warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
api-gateway-1 | No XML encryptor configured. Key {411d36c7-805a-49e9-a635-614b5a396591} may be persisted to storage in unencrypted form.
api-gateway-1 | warn: Microsoft.AspNetCore.Hosting.Diagnostics[15]
api-gateway-1 | Overriding HTTP_PORTS '8080' and HTTPS_PORTS ''. Binding to values defined by URLs instead 'http://*:80'.
api-gateway-1 | info: Microsoft.Hosting.Lifetime[14]
api-gateway-1 | Now listening on: http://[::]:80
api-gateway-1 | info: Microsoft.Hosting.Lifetime[0]
api-gateway-1 | Application started. Press Ctrl+C to shut down.
api-gateway-1 | info: Microsoft.Hosting.Lifetime[0]
api-gateway-1 | Hosting environment: Production
api-gateway-1 | info: Microsoft.Hosting.Lifetime[0]
api-gateway-1 | Content root path: /app
View in Docker Desktop View Config Enable Watch
```

Приложение запустилось. Можно протестировать по ссылке:

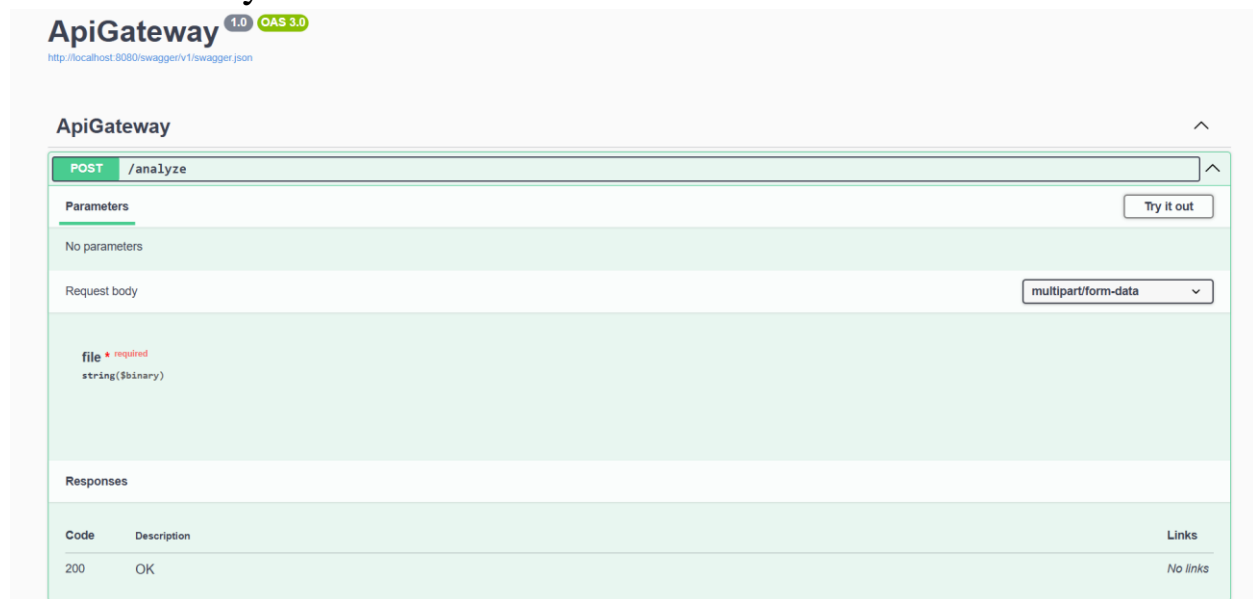
<http://localhost:8080/swagger>

Теперь загрузим файлы разного типа и проверим работоспособность программы.

После открытия ссылки видим следующее:



Нажимаем try it out.



Загрузим такой txt-файл:

Hello, World!

KPO

Software Design
BHW-2

FCS

Kaspari David

Нажимаем Execute. Получаем такой ответ.

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/analyze' \
  -H 'accept: */*' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file@papa.txt;type=text/plain'
```

Request URL

http://localhost:8080/analyze

Server response

| Code | Details |
|------|---|
| 200 | <p>Response body</p> <pre>{ "paragraphCount": 5, "wordCount": 10, "characterCount": 72, "analysisDate": "2025-05-21T23:37:12.453442Z" }</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Wed, 21 May 2025 23:37:11 GMT server: Kestrel transfer-encoding: chunked</pre> |

Responses

| Code | Description |
|------|-------------|
|------|-------------|

Links

Как видим, получили верный ответ с кол-вом символов, абзацев, слов, а также датой обращения.

Загрузим файл другого формата (наш word-отчет, например).

Curl

```
curl -X 'POST' \
  'http://localhost:8080/analyze' \
  -H 'accept: */*' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file@Kaspari_237_order.docx;type=application/vnd.openxmlformats-officedocument.wordprocessingml.document'
```

Request URL

http://localhost:8080/analyze

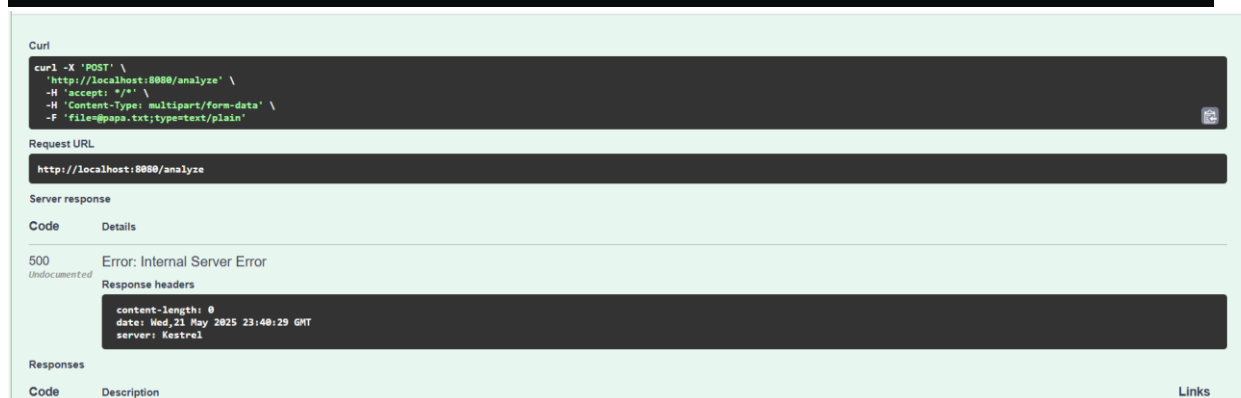
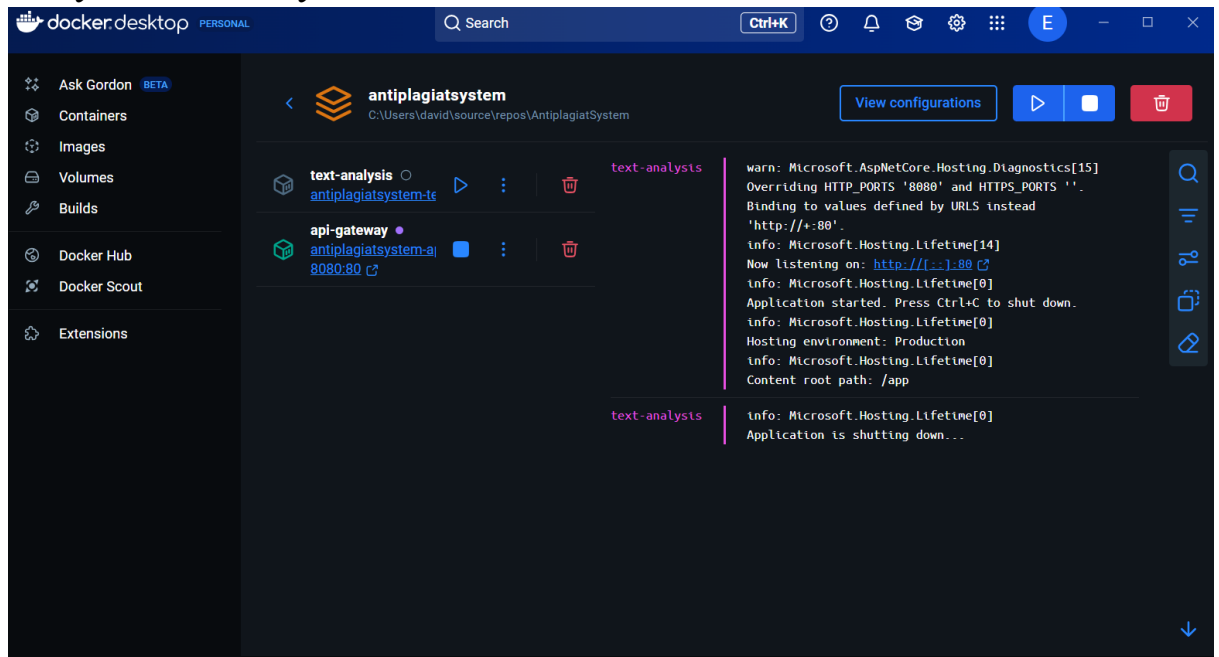
Server response

| Code | Details |
|------|--|
| 400 | <p>Error: Bad Request</p> <p>Response body</p> <pre>"Only .txt please"</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Wed, 21 May 2025 23:39:33 GMT server: Kestrel transfer-encoding: chunked</pre> |

Responses

Получили ответ, что принимаются только txt-файлы.

Если отключить один из серверов (TextAnalysis, например), то получим ошибку:



В целом, это вся работа. Пройдемся по критериям оценки, чтоб проверить соответствие данной работы заданию:

Основные требования к функциональности реализованы (так получилось, что это только первый пункт).

Серверная часть приложения состоит из двух микросервисов (ApiGateway и TextAnalysisService).

Если один из микросервисов не работает, то показывается ошибка.

Swagger демонстрирует функциональность реализованных микросервисов.

Код разделен на модули и имеет структуру, описанную выше. Документация (то есть этот отчет) содержит краткое описание архитектуры системы и спецификацию API.

Как видно, основные требования к программе выполнены.

В процессе работы я познакомился с тем, как работает синхронное межсервисное взаимодействие, поработал со Swagger'ом, а также с Docker.