

КР №2 по КПО.

Магомедов Абдул Омаргаджиевич

БПИ 234

Предисловие:

Комментарии в коде решил не писать, потому что думал, что тут понятнее объясню скринами. На github перенес проект с помощью Github Desktop, но т.к. там есть некоторые доп. папки, к примеру папки с тестовыми .txt файлами, я загрузил zip-архив, чтобы можно было отдельно скачать сам проект. КР сделано на оценку 10, вроде.

Краткое описание:

Вся архитектура проекта состоит из системы 3-ех компонентов:

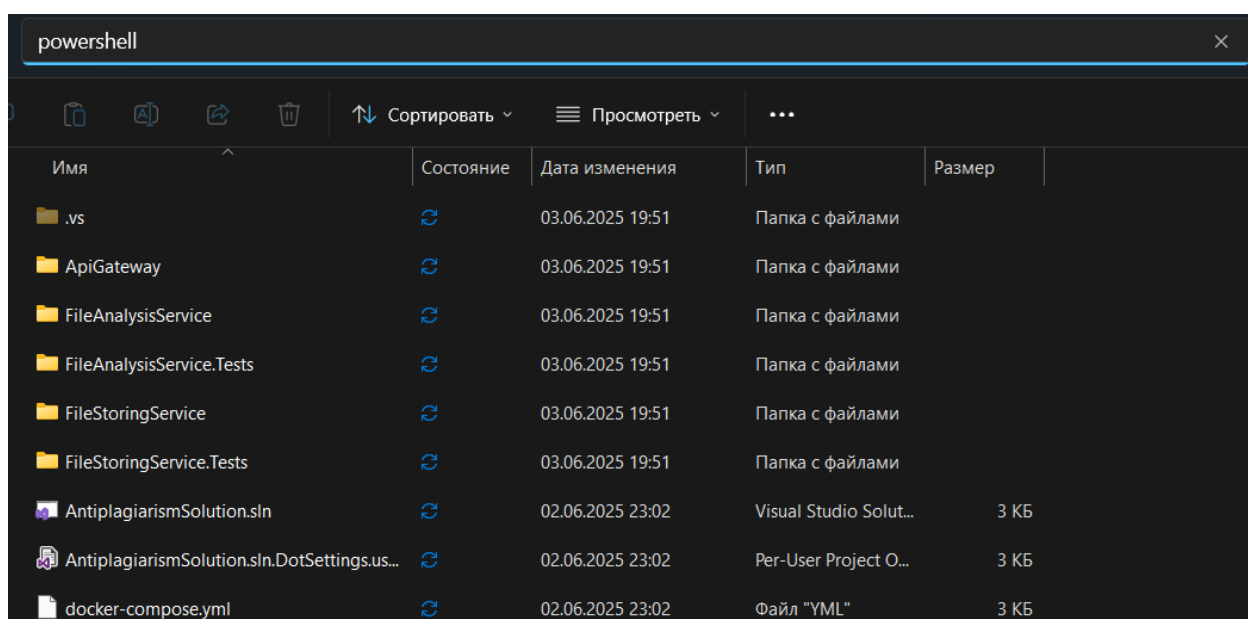
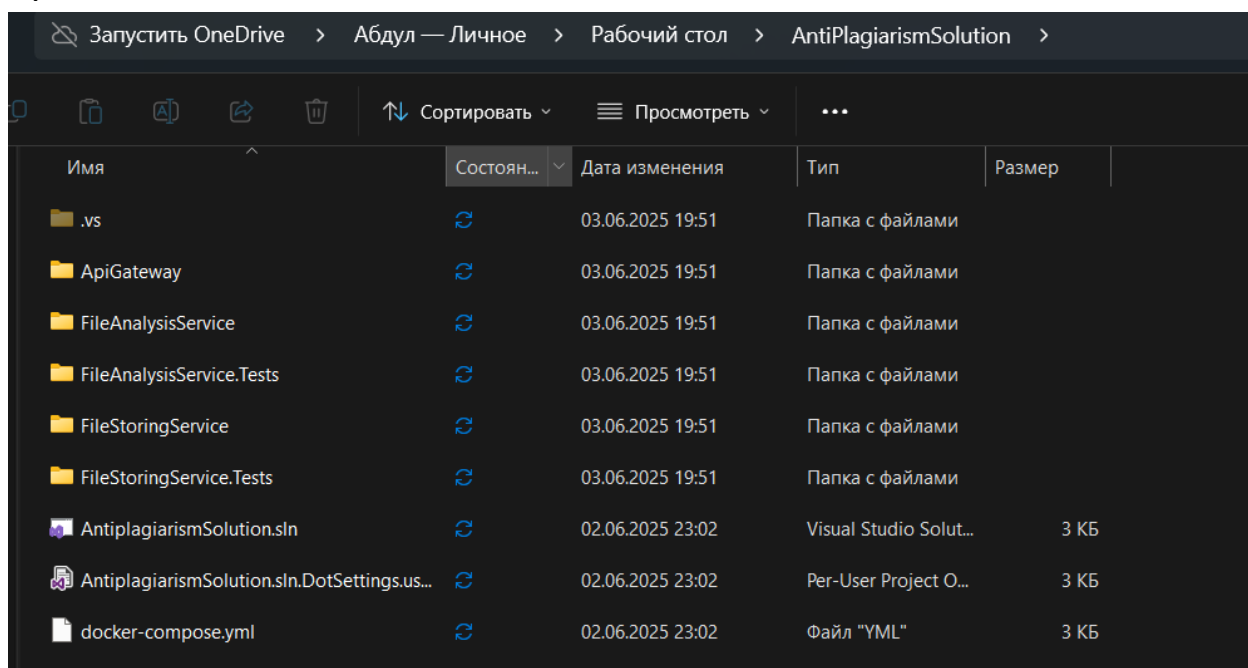
- API Gateway - Отвечает за маршрутизацию запросов между клиентом и двумя другими микросервисами.
- FileStoringService - Принимает и сохраняет файлы, а также предоставляет доступ для скачивания загруженных файлов.
- FileAnalysisService - Проводит анализ загруженного файла (подсчёт слов, символов и т.д.) и хранит результаты анализа.

Таким образом, API Gateway служит неким соединением всех микросервисов, однако по отдельности они работают точно так же и синхронно, то есть если я загрузил файл в API Gateway, то могу сделать анализ с помощью id загруженного файла уже на отдельном swagger'е FileAnalysisService, т.к файл записан в базу данных со своим ip.

Краткая инструкция:

Чтобы объединить все 3 микросервиса, я воспользовался Docker Desktop.

После скачивания zip-архива нужно запустить .sln файл, чтобы просмотреть саму структуру проекта ну и позже запустить, и проверить тесты, а так в начале можно и не заходить. Заходим в папку с проектом и в ячейке, где расположен путь файла вводим «powershell» и жмем Enter.

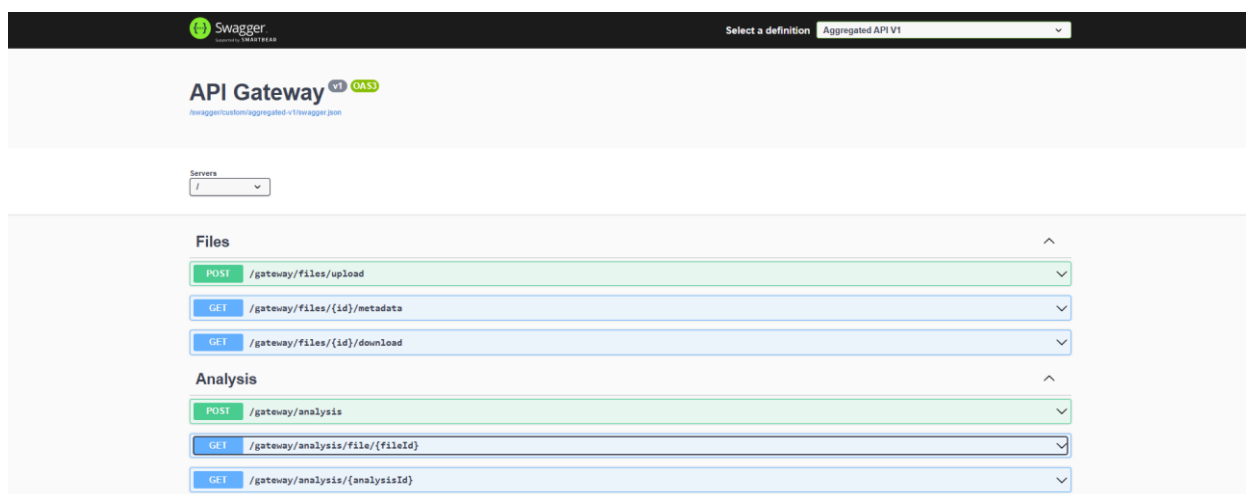


После чего у нас автоматически открывается powershell с нужным путем. Далее для запуска вводим обязательно такую команду: «docker-compose up --build». С другими командами у меня запустить не получилось. После окончания процесса, в Docker Desktop можно увидеть, что наш контейнер успешно запустился:

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	antiplagiarismsolution	-	-	-	0.92%	12 seconds ago	
<input type="checkbox"/>	db_fileanalysis-1	d36df6a363af	postgres:15-alpine	5434:5432 ↗	0.05%	13 seconds ago	
<input type="checkbox"/>	db_filestoring-1	6fc85c31fd71	postgres:15-alpine	5433:5432 ↗	0.03%	13 seconds ago	
<input type="checkbox"/>	filestoringservice-1	f514666cb3a2	antiplagiarismsolution-file: 8001:8080 ↗		0.35%	13 seconds ago	
<input type="checkbox"/>	fileanalysiservice-1	c66bde5d0732	antiplagiarismsolution-file: 8002:8080 ↗		0.35%	13 seconds ago	
<input type="checkbox"/>	apigateway-1	ea52f6c7f331	antiplagiarismsolution-api: 8000:8080 ↗		0.14%	12 seconds ago	

Чтобы перейти на swagger'ы, нужно в столбце Port(s) нажать на соответствующий порт, после чего в ссылке добавить “/swagger”, для удобства в README.md на github добавлю все действенные ссылки. Теперь перейдем к самому функционалу микросервисов.

По ссылкам можем перейти на 3 вышеупомянутых микросервиса, тут я буду рассматривать только API Gateway, т.к. как я уже сказал ранее, API Gateway по сути хранит в себе два других микросервиса, поэтому отдельно рассматривать их нет смысла, но можно проверить, чтобы удостовериться, что по отдельности они работают абсолютно так же.



В самом API Gateway мы видим 2 секции, Files и Analysis.

В ячейку POST upload мы должны загрузить текстовый файл, на котором будет произведен анализ:

Files

POST /gateway/files/upload

Parameters

No parameters

Request body

multipart/form-data

file

string(\$binary) Буфoр файла test.txt

☐ Send empty value

Execute Clear

После нажатия на “Execute” нам выдается некоторая информация о файле, из которых самая важная – это его id:

201

Response body

```
{
  "id": "6037834f-b145-4d93-a13d-fb75382d607d",
  "originalFileName": "test.txt",
  "contentType": "text/plain",
  "fileSize": 46,
  "uploadDate": "2025-06-03T18:00:50.6363851Z",
  "hash": "0b87067204ec0191e88fc352349ba7240c8c61b6b8a3532c13353da56214c137"
}
```

Download

Ниже можно увидеть ситуации, в которых могут произойти ошибки, условно на примере если вы вместо текстового файла загрузим png:

400

Error: Bad Request

Response body

only .txt files are allowed.

Download

Далее, копируем id, в ячейке metadata можем ввести id файла, чтобы просмотреть информацию, которую описывает наш файл:

GET /gateway/files/{id}/metadata

Parameters

Name Description

id * required string(\$uuid) (path) 6037834f-b145-4d93-a13d-fb75382d607d

Execute Clear

200

Response body

```
{
  "id": "6037834f-b145-4d93-a13d-fb75382d607d",
  "originalFileName": "test.txt",
  "contentType": "text/plain",
  "fileSize": 46,
  "uploadDate": "2025-06-03T18:00:50.6363851Z",
  "hash": "0b87067204ec0191e88fc352349ba7240c8c61b6b8a3532c13353da56214c137"
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Tue, 03 Jun 2025 18:04:32 GMT
server: Kestrel
transfer-encoding: chunked
```

В условии сказано так:

«File Storing Service – отвечает только за хранение и выдачу файлов»

Так как указана выдача файлов, то в микросервисе FileStoringService так же реализована функция загрузки файла, так же по id в ячейке download:

GET /gateway/files/{id}/download

Parameters

Name	Description
id * required string(\$uuid) (path)	6037834f-b145-4d93-a13d-fb75382d607d

Execute Clear

Code Details

200

Response body
[Download file](#)

Response headers

```
content-disposition: attachment; filename=test.txt; filename*=UTF-8''test.txt
content-length: 46
content-type: text/plain
date: Tue, 03 Jun 2025 18:08:03 GMT
server: Kestrel
```

Теперь перейдем к секции **Analysis**.

Чтобы провести анализ, нам нужно перейти в ячейку POST analysis, где нам так же понадобится id файла:

Analysis

POST /gateway/analysis

Parameters

No parameters

Request body

application/json

```
{
  "fileId": "6037834f-b145-4d93-a13d-fb75382d607d",
  "fileHash": "string",
  "forceReanalyze": true
}
```

Execute Clear

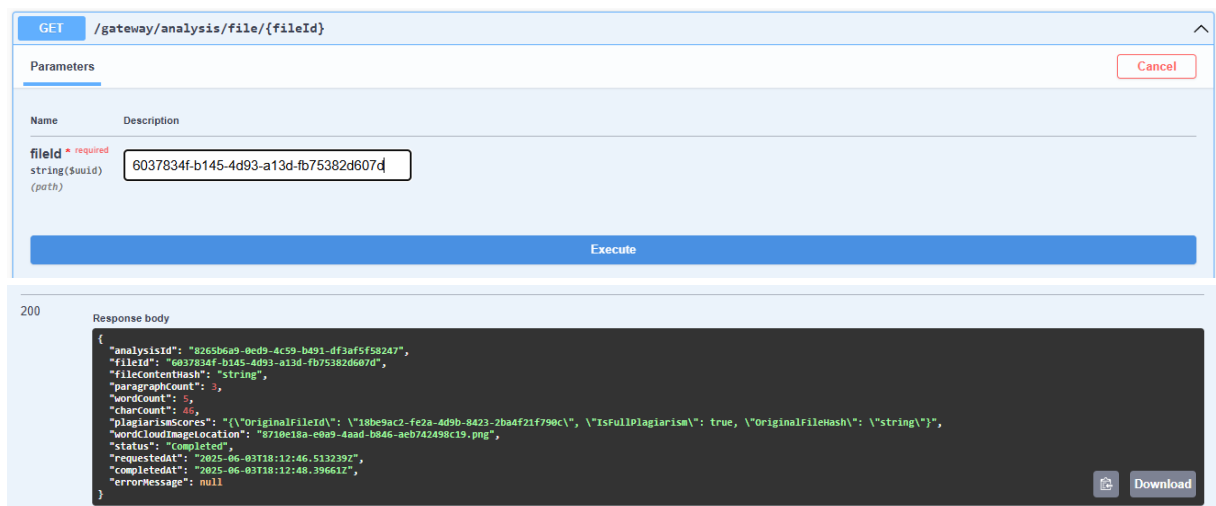
Тут мы видим следующую информацию:



```
{
  "analysisId": "8265b6a9-0ed9-4c59-b491-df3af5f58247",
  "fileId": "6037834f-b145-4d93-a13d-fb75382d607d",
  "fileContentHash": "string",
  "paragraphCount": 3,
  "wordCount": 5,
  "charCount": 46,
  "plagiarismScores": "{\\\"OriginalFileId\\\": \\\"18be9ac2-fe2a-4d9b-8423-2ba4f21f790c\\\", \\\"IsFullPlagiarism\\\": true, \\\"OriginalFileHash\\\": \\\"string\\\"}",
  "wordCloudImageLocation": "8710e18a-e0a9-4aad-b846-acb742498c19.png",
  "status": "Completed",
  "requestedAt": "2025-06-03T18:12:46.513239Z",
  "completedAt": "2025-06-03T18:12:48.396610Z",
  "errorMessage": null
}
```

Мы видим весь реализованный анализ, paragraphCount, wordcount, charCount. Так же помимо id загружаемого файла, нам выводит id реализованного анализа. Это нужно для того, чтобы в дальнейшем проверить, был ли совершен анализ и действительно ли работают бд так, как надо.

Перейдем к проверке анализа через fileId:



```
GET /gateway/analysis/file/{fileId}

Parameters

Name      Description
fileId * required
string($uuid)
(path)    6037834f-b145-4d93-a13d-fb75382d607d

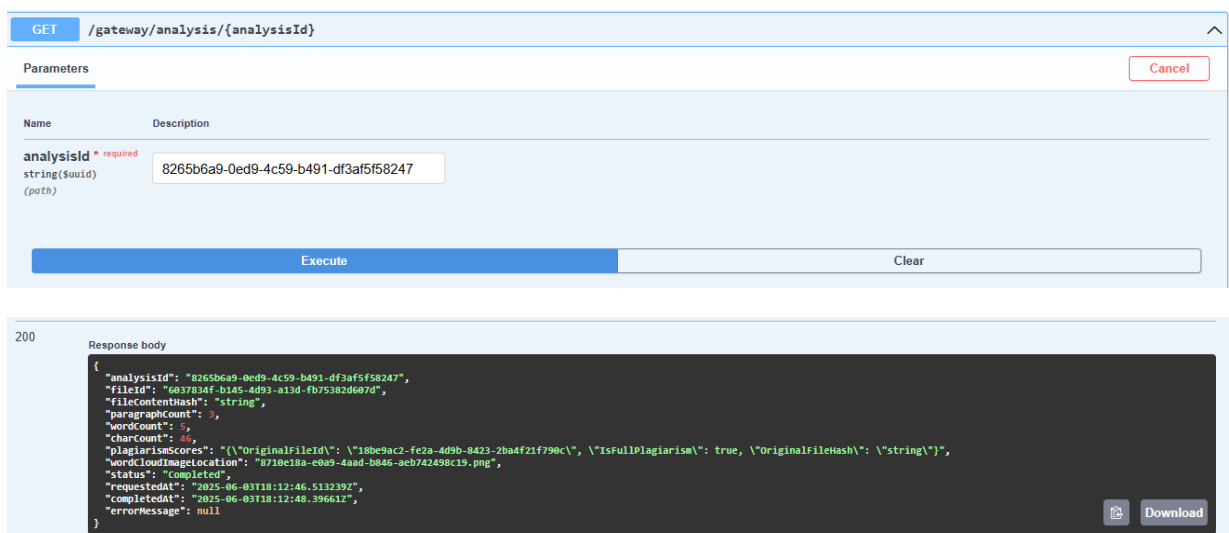
Execute

200

Response body

{
  "analysisId": "8265b6a9-0ed9-4c59-b491-df3af5f58247",
  "fileId": "6037834f-b145-4d93-a13d-fb75382d607d",
  "fileContentHash": "string",
  "paragraphCount": 3,
  "wordCount": 5,
  "charCount": 46,
  "plagiarismScores": "{\\\"OriginalFileId\\\": \\\"18be9ac2-fe2a-4d9b-8423-2ba4f21f790c\\\", \\\"IsFullPlagiarism\\\": true, \\\"OriginalFileHash\\\": \\\"string\\\"}",
  "wordCloudImageLocation": "8710e18a-e0a9-4aad-b846-acb742498c19.png",
  "status": "Completed",
  "requestedAt": "2025-06-03T18:12:46.513239Z",
  "completedAt": "2025-06-03T18:12:48.396612Z",
  "errorMessage": null
}
```

Как мы видим, выводит абсолютно ту же информацию, что говорит о том, что анализ был совершен и все хорошо работает. То же самое будет, если провести это с analysisId:



```
GET /gateway/analysis/{analysisId}

Parameters

Name      Description
analysisId * required
string($uuid)
(path)    8265b6a9-0ed9-4c59-b491-df3af5f58247

Execute

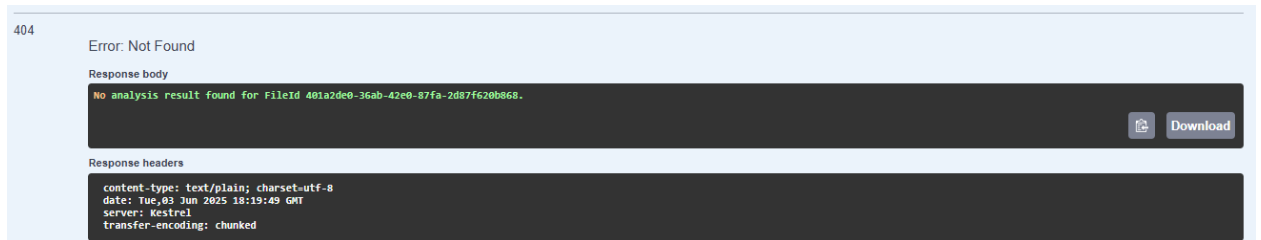
200

Response body

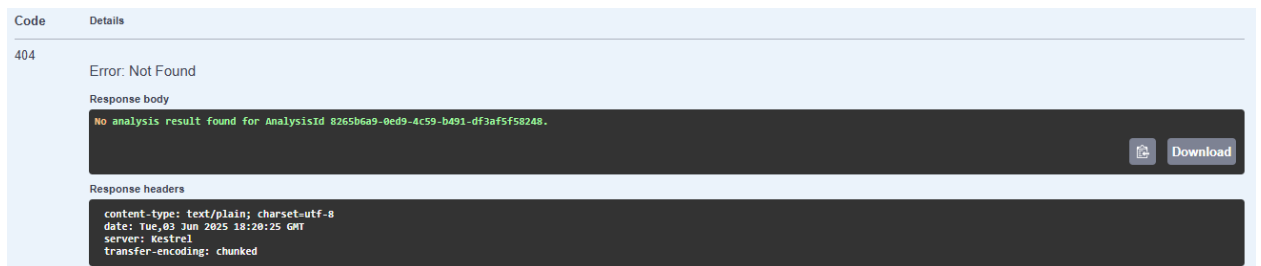
{
  "analysisId": "8265b6a9-0ed9-4c59-b491-df3af5f58247",
  "fileId": "6037834f-b145-4d93-a13d-fb75382d607d",
  "fileContentHash": "string",
  "paragraphCount": 3,
  "wordCount": 5,
  "charCount": 46,
  "plagiarismScores": "{\\\"OriginalFileId\\\": \\\"18be9ac2-fe2a-4d9b-8423-2ba4f21f790c\\\", \\\"IsFullPlagiarism\\\": true, \\\"OriginalFileHash\\\": \\\"string\\\"}",
  "wordCloudImageLocation": "8710e18a-e0a9-4aad-b846-acb742498c19.png",
  "status": "Completed",
  "requestedAt": "2025-06-03T18:12:46.513239Z",
  "completedAt": "2025-06-03T18:12:48.396612Z",
  "errorMessage": null
}
```

Так же как и в секции Files, снизу есть виды возможных ошибок, условно если мы загрузим файл, скопируем id и попробуем проверить, был ли совершен анализ через это id, при этом не анализируя сам файл, в данном случае нам выведет сообщение о том, что анализ файла еще не был совершен:

fileId:

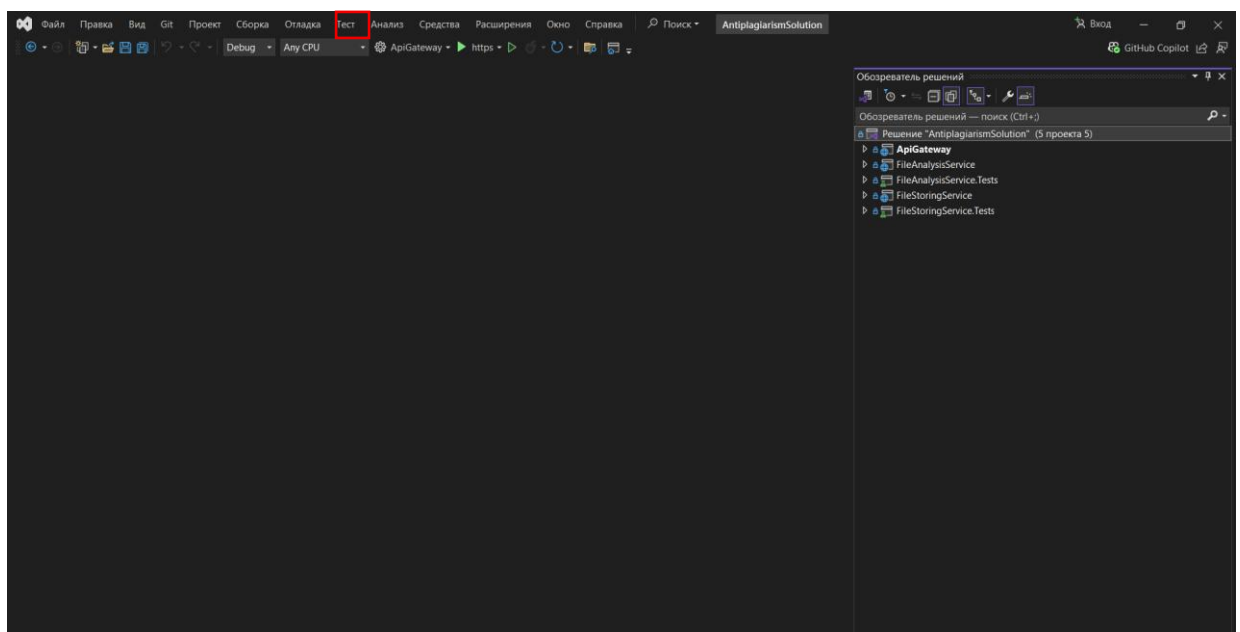


AnalysisId:

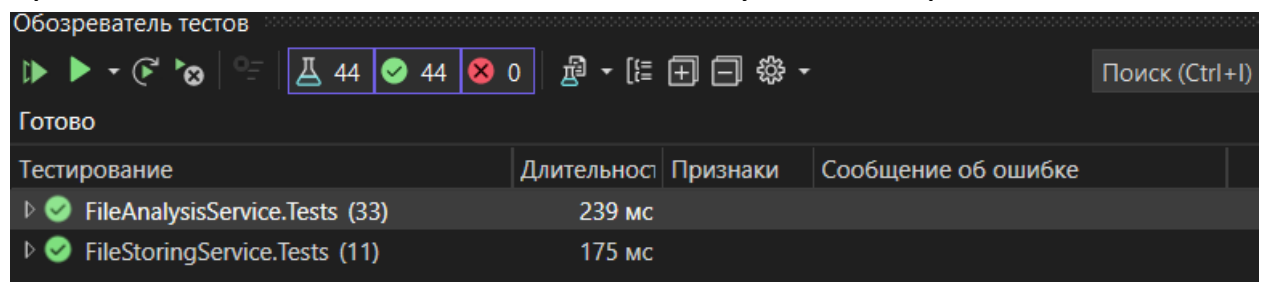


Покрытие тестами:

Запустив проект, в панели управления сверху выбираем ячейку «Тест»:

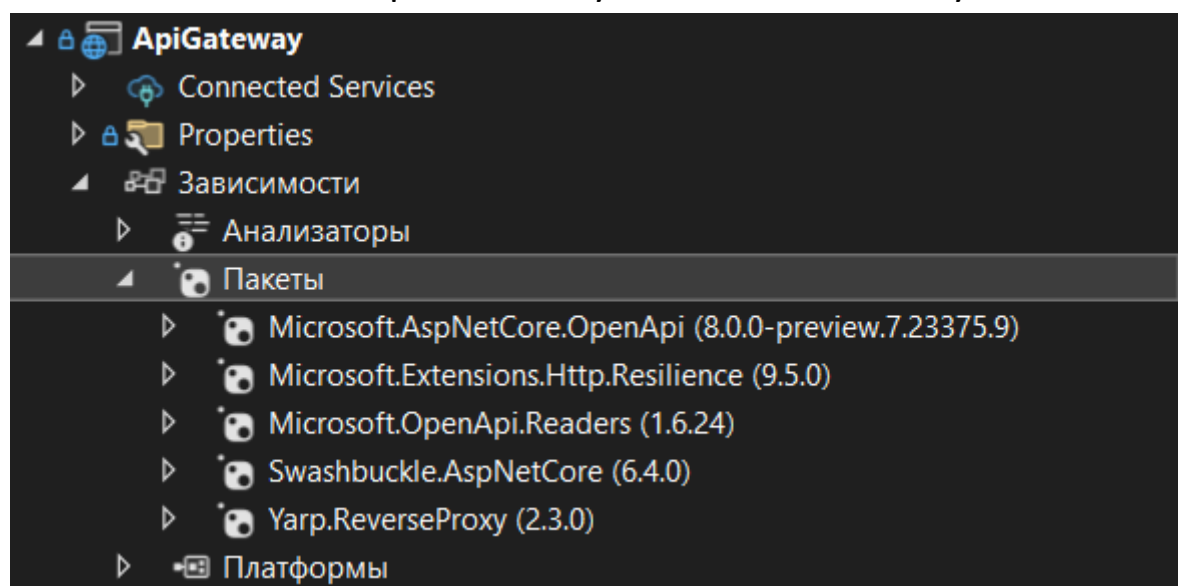


После чего выбираем «Запуск всех тестов», и через некоторое время нам покажет, что все тесты были успешно пройдены:



Nuget-пакеты:

дополнительно скачивать ничего не требуется, так как при скачивании самого проекта все уже автоматически установлено:



Вроде все.