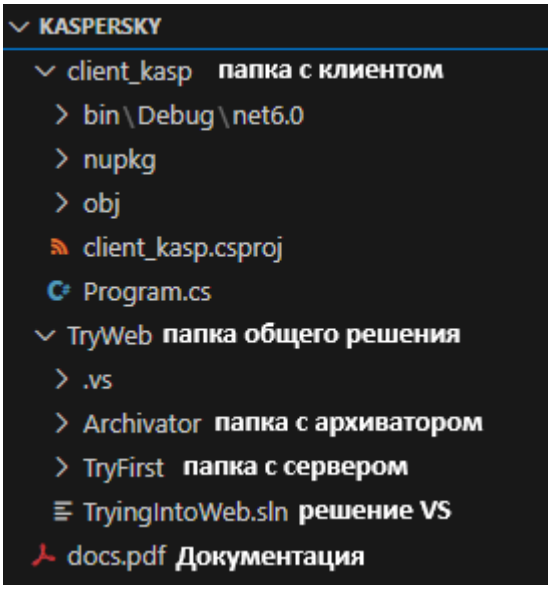
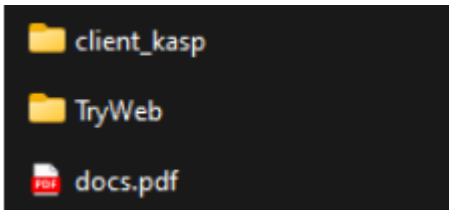
Разработка C#. SafeBoard H1 2024
Горин Никита
Документация.

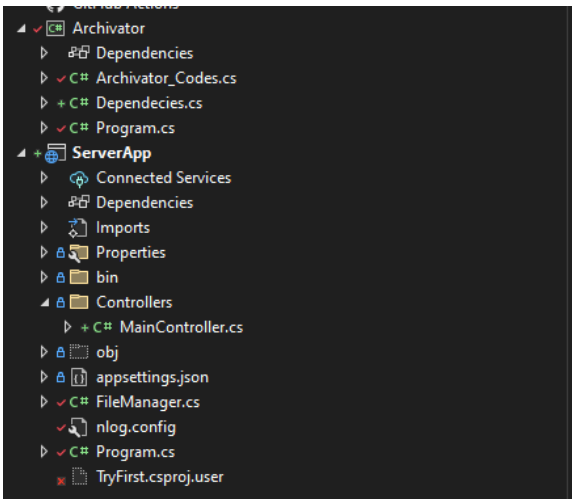
Реализовано три приложения, выступающих сервером, архиватором и клиентом.

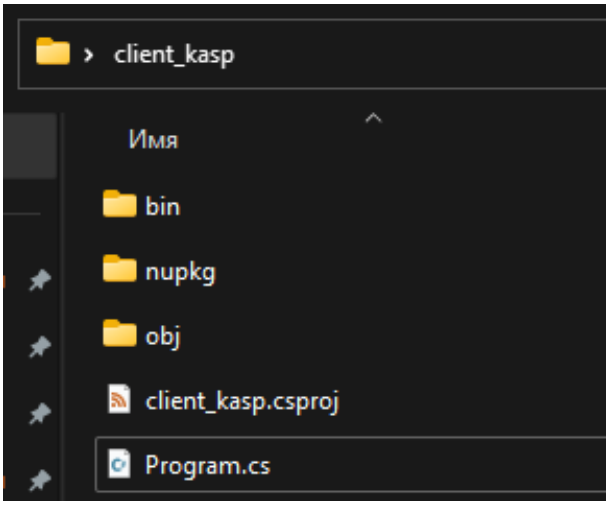
Структура папки





Структура сервера и архиватора (Solution, 2 projects)

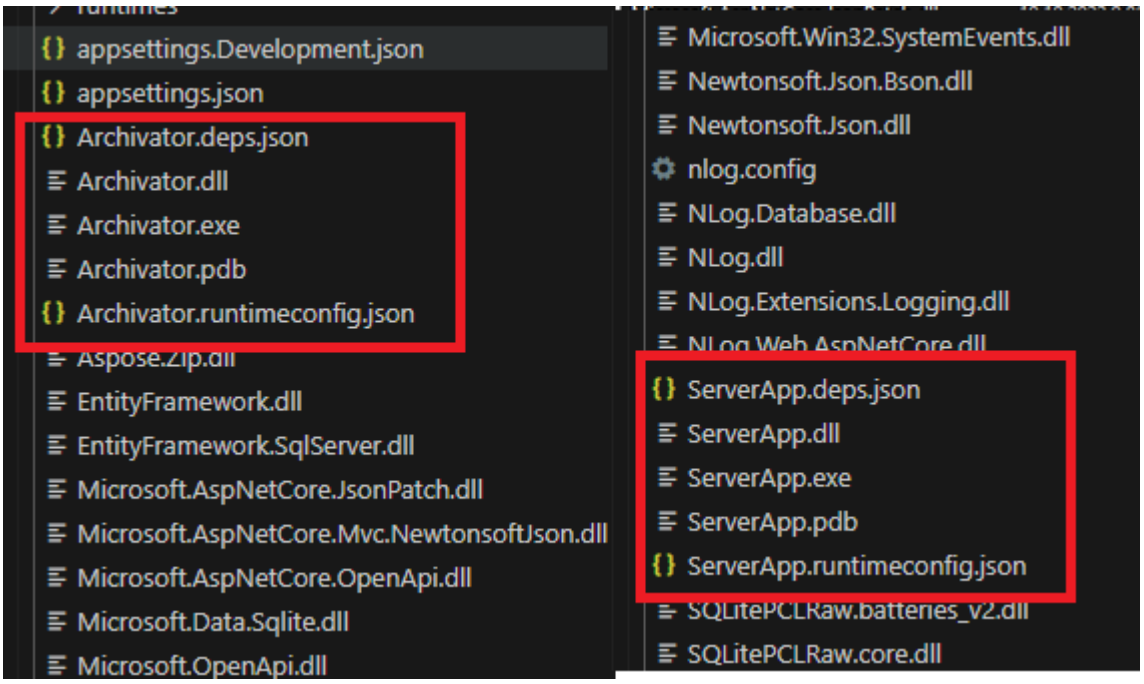




Структура клиента (Project)

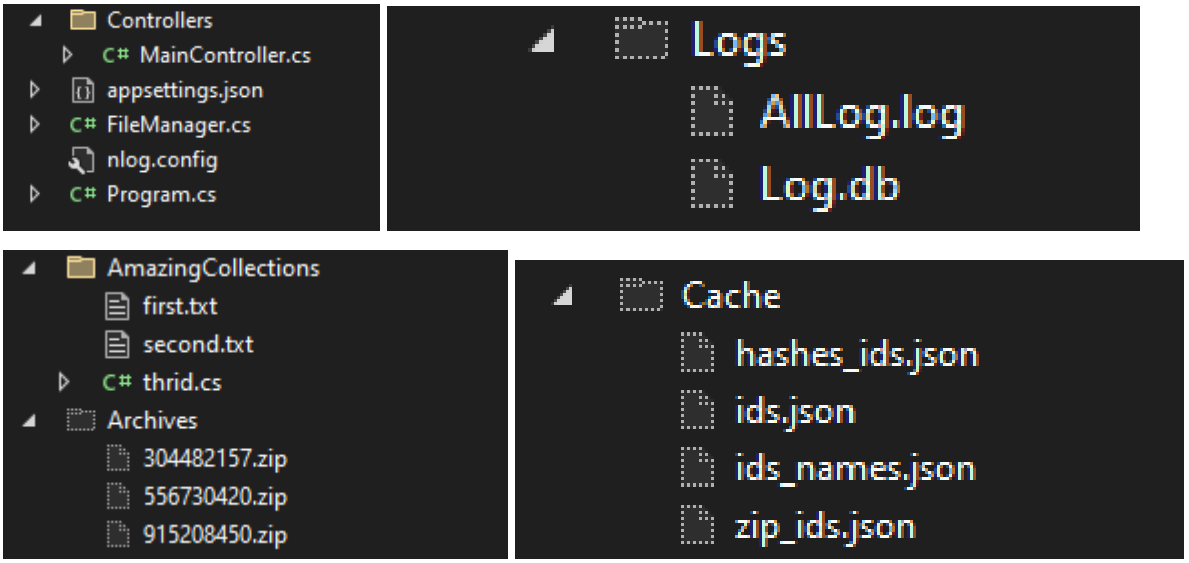
Сервер

Является ASP.NET Core 7.0 приложением, реализующим контроллер для RESTful HTTP сервиса. Во время сборки автоматически собирается и зависимый архиватор. Ниже на скриншоте показаны два соответствующих .exe файла.



Классы и важные папки

Состоит из api контроллера **MainController.cs**, файлового менеджера (**FileManager.cs**), настраивающего логи в базу данных и работу с кэшем, и основного **Program.cs** с билдом приложения.



В папке **AmazingCollections** хранятся самые удивительные файлы(!), архивация которых и происходит.

В папке **Archives** хранятся созданные архивы с уникальными id, за которыми следит **FileManager** и которые связаны с файлами из **Cache**.

В папке **Cache** хранятся 4 файла

1) **ids_names.json** - каждому файлу в Amazing Collections присвоен уникальный Id, в этом файле хранится Словарь, который соединяет имена файлов со своими Id.
2) **ids.json** - коллекция всех существующих Id файлов
3) **hashes_ids.json** - каждому списку файлов/архиву соответствует свой уникальный Id (который и входит в название файла). Здесь находится словарь, хранящий пары “список Id файлов - Id архива”
4) **zip_ids.json** - коллекция всевозможных Id архивов.

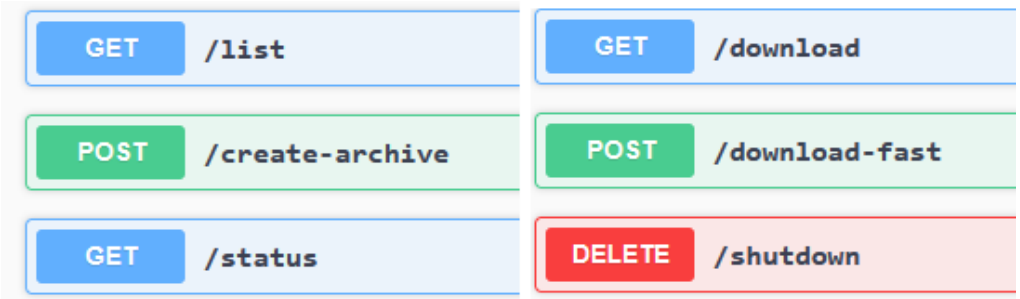
Все эти сложности были созданы для того, чтобы избежать линейной сложности различных проверок на уникальность, наличие и состояние каждого архива. **Dictionary** и **HashSet** позволяет ускорить алгоритмы прохода по до O(1).

Логи

Логгирование происходит с помощью пакета Nlog, где настроены логи в консоль и в файлы папки Logs (создается автоматически): обычный .log файл и простейшая одна таблица SQLite. Пример логов в таблице:

2024/05/22 20:03:33.646	Ok	Info		<no type>.lambda_method5 ...	ServerApp.Controllers.MainCo...	http://localhost/create-archive
2024/05/22 20:03:34.050	Process done	Info		Process.CompletionCallback =...	ServerApp.Controllers.MainCo...	
2024/05/22 20:03:38.679	Not Found processId	Error		<no type>.lambda_method8 ...	ServerApp.Controllers.MainCo...	http://localhost/status
2024/05/22 20:03:49.176	Success	Info		<no type>.lambda_method8 ...	ServerApp.Controllers.MainCo...	http://localhost/status
2024/05/22 20:04:10.571	Success download	Info		AsyncStateMachineBox`1.Mov...	ServerApp.Controllers.MainCo...	http://localhost/download
2024/05/22 20:04:36.950	Ok	Info		AsyncMethodBuilderCore.Start...	ServerApp.Controllers.MainCo...	http://localhost/download-fast
2024/05/22 20:04:36.987	Process in progress...	Info		AsyncMethodBuilderCore.Start...	ServerApp.Controllers.MainCo...	http://localhost/download-fast
2024/05/22 20:04:37.042	Process in progress...	Info		AsyncMethodBuilderCore.Start...	ServerApp.Controllers.MainCo...	http://localhost/download-fast
2024/05/22 20:04:37.105	Process in progress...	Info		AsyncMethodBuilderCore.Start...	ServerApp.Controllers.MainCo...	http://localhost/download-fast
2024/05/22 20:04:37.167	Process in progress...	Info		AsyncMethodBuilderCore.Start...	ServerApp.Controllers.MainCo...	http://localhost/download-fast
2024/05/22 20:04:37.230	Process in progress...	Info		AsyncMethodBuilderCore.Start...	ServerApp.Controllers.MainCo...	http://localhost/download-fast
2024/05/22 20:04:37.293	Process in progress...	Info		AsyncMethodBuilderCore.Start...	ServerApp.Controllers.MainCo...	http://localhost/download-fast
2024/05/22 20:04:37.356	Process in progress...	Info		AsyncMethodBuilderCore.Start...	ServerApp.Controllers.MainCo...	http://localhost/download-fast
2024/05/22 20:04:37.390	Process done	Info		Process.CompletionCallback =...	ServerApp.Controllers.MainCo...	http://localhost/download-fast
2024/05/22 20:04:37.419	Success	Info		AsyncMethodBuilderCore.Start...	ServerApp.Controllers.MainCo...	http://localhost/download-fast
2024/05/22 20:04:37.419	Success download	Info		AsyncStateMachineBox`1.Mov...	ServerApp.Controllers.MainCo...	http://localhost/download-fast
2024/05/22 20:05:03.962	Shutting down...	Info		SyncActionResultExecutor.Exec...	ServerApp.Controllers.MainCo...	http://localhost/shutdown
2024/05/22 20:05:41.380	Ok	Info		<no type>.lambda_method6 ...	ServerApp.Controllers.MainCo...	http://localhost/create-archive
2024/05/22 20:06:03.694	Success download	Info		AsyncStateMachineBox`1.Mov...	ServerApp.Controllers.MainCo...	http://localhost/download
2024/05/22 20:06:29.044	Not Found processId	Error		<no type>.lambda_method18 ...	ServerApp.Controllers.MainCo...	http://localhost/status
2024/05/22 20:06:41.239	Ok	Info		<no type>.lambda_method6 ...	ServerApp.Controllers.MainCo...	http://localhost/create-archive
2024/05/22 20:06:41.685	Process done	Info		Process.CompletionCallback =...	ServerApp.Controllers.MainCo...	

Эндпоинты



/list - GET

Получает список всех файлов, хранящихся в **AmazingCollections** и выводит их клиенту (подразумевается отсутствие вложенных папок)

```
//Get method.
//Empty Body
//0 Parameters
//Returns a list of available Files in AmazingCollection
[HttpGet("/list")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
```

/create-archive - POST

Создает процесс **Archivator.exe**, заносит в текущий словарь процессов уникальный Id нового процесса, подписывается на его завершение (чтобы узнать статус код) и возвращает этот Id

В случае, если список файлов повторяется (кэш), то также создает уникальный Id с приставкой Cache, возвращает этот Id

0 параметров
Из тела реквеста получает список строк, соответствующий названиям файлов для архивации

```
//Post method.
//Body - {file1, file2, file3...} (in json)
//0 Parameters
//Creates a process of archivating files
[HttpPost("/create-archive")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
0 references
```

/status - GET

Проверяет статус процесса и при его присутствии возвращает один из **ExitCode** архиватора - **Success, UserError, ServerError, Calculating**.
1 параметр - Id процесса

```
//Get method
//Empty Body
//1 parameter - processIdStr
//only digits (e.g 429428942, 298290342) - recently created new processes
//Cache******* (e.g Cache234829924, Cache32748382) - Already in Cache -> Instantly Success
//returns ArchCode (Calculating, ServerError, UserError or Success)
[HttpGet("/status")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
0 references
```

/download - GET

Скачивает архив, возвращая в Content поток, из которого клиент может создать полноценный файл (указав путь).

1 параметр - Id процесса, связанный с которым архив необходимо скачать.

Замечание: Статус Код этого процесса должен быть Success, иначе загрузка будет невозможна.

```
//Get method
//Empty body
//1 parameter - processIdStr
//returns File.
[HttpGet("/download")]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
1 reference
public async Task<IActionResult> Download(string processIdStr)
{
```

/download-fast - POST

Последовательно выполняет /create-archive, /status, /download, полностью скрывая от пользователя уникальные Id процессов и архивов.

Принимает из тела реквеста имена файлов из AmazingCollections, и возвращает поток на скачивание архива, состоящего из этих файлов.

```
//Post method
//Empty body
//1 parameter - processIdStr
//returns File. Directly from FileList
[HttpPost("/download-fast")]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
0 references
public async Task<IActionResult> Download_Real([FromBody] List<string> filenames)
{
```

/shutdown - DELETE

Останавливает работу сервера и кэширует все архивы, созданные за время его работы.

```
//Only right method to save Cache
[HttpDelete("/shutdown")]
[ProducesResponseType(StatusCodes.Status204NoContent)]
0 references
public IActionResult ShutDown()
{
```

Архиватор

Незамысловатое консольное приложение, запускаемое сервером и получаемое на вход аргументы:

(processId, basedirectory, zip_name(id), file1, file2, …)

- **processId** - уникальный Id этого процесса
- **zip_name** - уникальный Id архива
- **basedirectory** - директория, в которой запущено приложение (сервера!)
- **file1, file2,...** - имена файлов, которые будут в архиве

Максимальный размер файла - **512 КБ**.

Возвращает один из статус кодов:

```
namespace Archivator
{
    public enum ArchCodes
    {
        Success = 0,
        UserError = 1,
        ServerError = 2,
        Calculating = 3
    }
}
```

```
nst string IDS_PATH = "Cache/ids.json";//hashset<i
nst string NAMES_IDS_PATH = "Cache/ids_names.json"

nst string ZIP_IDS = "Cache/zip_ids.json";//hashse
nst string ZIP_HASHES = "Cache/hashes_ids.json";//
nst string ARCHIVATOR_PATH = "Archivator.exe";

nst string DATABASE_PATH = "Logs/Log.db";

nst string DATABASE_COMMAND = "CREATE TABLE Log ("
CreatedOn TEXT," +
Message TEXT " +
```

Примечание: **Calculating** никогда архиватор не вернет, это статус код, который привязывается к процессу сервером, и именно его заменяют остальные статус коды.

Кроме Program.cs и Archivator_Codes.cs содержит также и файл Dependencies.cs, который хранит к папкам и важным файлам. Этой информацией пользуются как сервер, так и архиватор.

Клиент

Примитивное консольное приложение, вся логика которого заключается в отправке вышеописанных запросов на на сервер. net core 7.0

Пакуется в утилиту. Ниже скриншот:

```
PS E:\GitHub\KASPERSKY\client_kasp> dotnet build   сборка
Версия MSBuild 17.7.1+971bf70db для .NET
  Определение проектов для восстановления...
  Все проекты обновлены для восстановления.
  client_kasp -> E:\GitHub\KASPERSKY\client_kasp\bin\Debug\net6.0\client_kasp.dll

Сборка успешно завершена.
    Предупреждений: 0
    Ошибок: 0
```

```
Прошло времени 00:00:00.69          локальная установка утилиты
PS E:\GitHub\KASPERSKY\client_kasp> dotnet tool install --global --add-source ./nupkg client_kasp
Вы можете вызвать это средство с помощью следующей команды: client_kasp
Средство "client_kasp" (версии "1.0.0") успешно установлено.
PS E:\GitHub\KASPERSKY\client_kasp> client_kasp команда, запускающая
Client app started...                          приложения
Type <exit> to exit
Use <help> for help
> exit
Exiting...                                                  деинсталляция
PS E:\GitHub\KASPERSKY\client_kasp> dotnet tool uninstall client_kasp --global утилиты
Инструмент "client_kasp" (версия "1.0.0") удален.
PS E:\GitHub\KASPERSKY\client_kasp>
```

Интерфейс взаимодействия клиента с сервером.

Unit-тесты

Я руками много тестировал, честно… Coming soon…

Publish

До докера руки не дошли :/

P.S. 32 МБ - мало для отправки файлов с этими проектами :/