Решение следующей задачи

1. WCF-сервис

**Реализует 2 метода:**

1. Принимает чек и сохраняет в БД.
2. Возвращает N-последних чеков из БД.

**Interface репозитория для общения с БД:**

1. Принимает чек и сохраняет в БД.
2. Возвращает N-последних чеков из БД.

**Основная реализация интерфейса репозитория для общения с БД.**  
 *[custom].[save\_cheques] (1й метод)*Входные параметры:

|  |  |  |
| --- | --- | --- |
| Параметр | Тип | Примечение |
| cheque\_id | uniqueidentifier |  |
| cheque\_number | nvarchar(50) |  |
| summ | money |  |
| discount | money |  |
| articles | nvarchar(max) | артикли чека, сджойненные символом “;”. |

*[custom].[get\_cheques\_pack] (2й метод)*Входные параметры:

|  |  |  |
| --- | --- | --- |
| Параметр | Тип | Примечение |
| pack\_size | integer |  |

Возвращаемые данные:   
коллекция объектов модели чека.

***Фейковая реализация интерфейса репозитория для общения с БД:***

*1 метод принимает чек и сохраняет его в App\_Data.  
2 метод возвращает N чеков с рандомным ID и номером от 1 до N (остальные поля по-умолчанию).  
  
Есть возможность указать в конфиге сервиса какую реализацию репозитория использовать.*

1. Windows-сервис
2. Мониторит папку на событие появление нового файла \*.txt файла (путь к папке указывается в конфиге).
3. Файл в себе должен содежать JSON чека.
   1. Если в папке оказывается некорректный файл (не верное расширение, не десериализуется, etc), то он перемещается в папку Garbage (путь указывается в конфиге).
4. Если файл корректный, то чек отправлется в 1й метод WCF-сервиса.
5. Файл перемещается в папку Complete
6. Пути к Garbage и Complete могут быть и относительными, и абсолютными.

Модель чека

* Guid **Id**
* string **Number**
* decimal **Summ**
* decimal **Discount**
* string[] **Articles**

Требования

* Общение с БД через Dapper, БД MsSQL.
* Работа с JSON через NewtonSoft.JSON.
* Логирование через log4net.

Порядок использования решения:

В решении пять проектов. Два основных - wcf (ConsoleAppServiceHost) и windows служба (WindowsServiceFolderMonitor)  и три вспомогательных - dll сборки, объекты которых используются в двух основных проектах.Wcf служба для простоты хостится в консольном приложении. Ведётся логирование.

Порядок использования:

0. Клонируем/скачиваем с git hub <https://github.com/proskurin-vp/WcfService> решение. Пересобираем решение (использовалась Visual Studio 2017) - подтянутся необходимые библиотеки.

1. Запускаем wcf службу (от имени администратора).

2. Устанавливаем windows службу. Для этого можно использовать утилиту InstallUtil.exe.

3. Запускаем windows службу. Она создаст папки Monitor, Complite, Garbage. Пути к ним берутся из файла конфигурации.

Реализовано два репозитория. Какой используется, указывается в файле конфигурации. Оба рабочие. Один пишет данные чека в таблицу базы данных, другой просто в папку, путь к которой указывается в конфигурационном файле.

4. При перемещении в папку Monitor "правильного" файла чека его данные методом wcf службы запишутся в бд или файл (зависит от настроек в файле конфигурации) и он будет Windows службой перемещён в папку Complite. Если будет в папку Monitor перемещён "неправильный" файл, он будет перемещён в папку Garbage. Если имя файла будет совпадать с уже имеющимися в папках  Complite или Garbage, к имени файла допишется метка времени.

База данных локальная, в папке bin, подключение через connectionString. Запись в таблицу выполняет хранимая процедура через dapper.