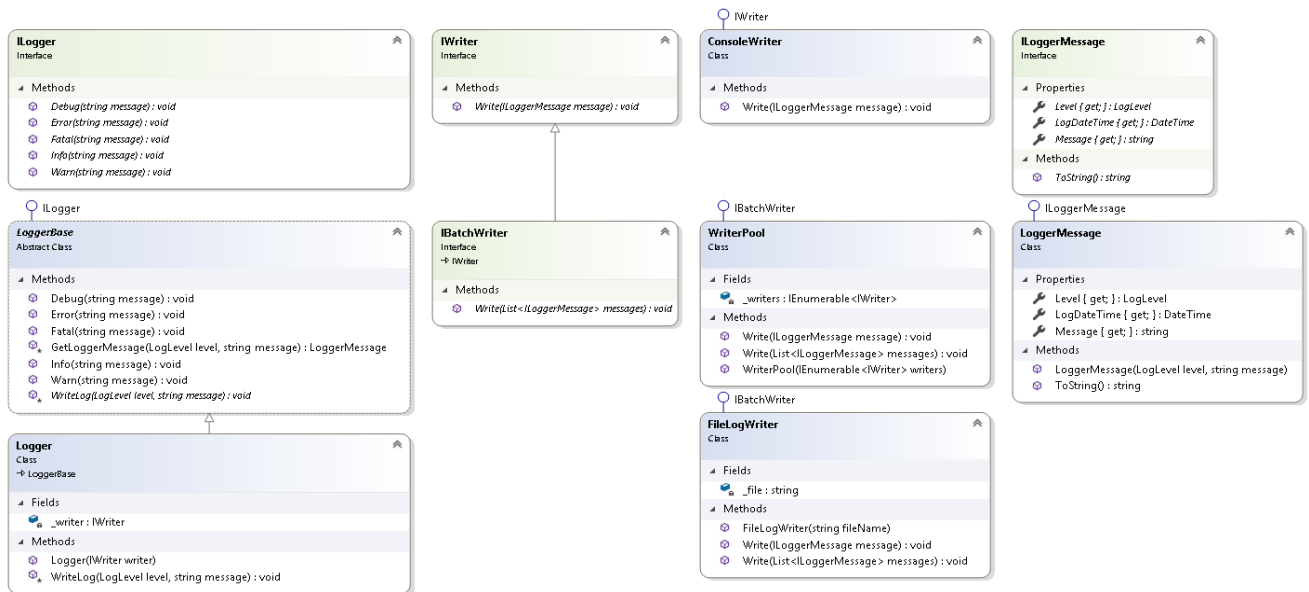


## Тестовое задание на позицию .NET Developer, вариант 1

### 1. Разработать библиотеку логирования событий (далее - Logger)

1.1 Logger должен реализовывать следующую схему классов (схема может быть аргументированно дополнена/изменена):



1.2 Logger должен поддерживать запись лога в различные хранилища, изменение целевого хранилища должно осуществляться инжектированием соответствующего writer-a;

1.3 Logger должен поддерживать одновременную запись в несколько хранилищ данных (например консоль, текстовый файл, БД, etc) - для тестового достаточно одновременно записи в консоль и текстовый файл;

1.4 Опционально: Logger должен поддерживать буферизацию сообщений в память и запись всего буфера команд в хранилище по команде или условию;

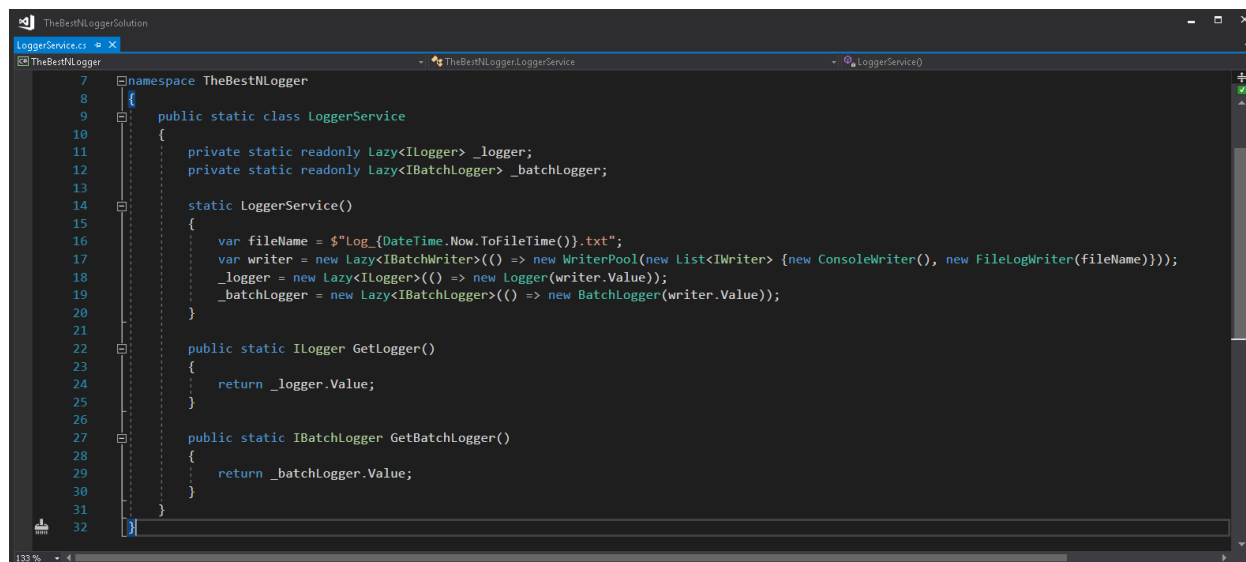
2. Создать консольное приложение с примерами использования логера;

2.1 В консольном приложении мы ожидаем увидеть примеры выхода логгера для различных ситуаций, например:

```
TheBestNLoggerSolution
Programs
  TheBestNLogger.Example
  TheBestNLogger.Console.Program
  Main0

1 namespace TheBestNLogger.Console
2 {
3     internal class Program
4     {
5         private static void Main()
6         {
7             //Example simple logger
8             var logger = LoggerService.GetLogger();
9             logger.Info("Simple logger: Info message");
10            logger.Warn("Simple logger: Warn message");
11            logger.Error("Simple logger: Error message");
12            logger.Debug("Simple logger: Debug message");
13            logger.Fatal("Simple logger: Fatal message");
14
15            //Example batch logger
16            var batchLogger = LoggerService.GetBatchLogger();
17            batchLogger.Info("Batch logger: Info message");
18            batchLogger.Warn("Batch logger: Info message");
19            batchLogger.Error("Batch logger: Error message");
20            batchLogger.Debug("Batch logger: Debug message");
21            batchLogger.Fatal("Batch logger: Fatal message");
22            batchLogger.Flush();
23        }
24    }
25 }
```

Пример инициализации логгера:



```
7 namespace TheBestNLogger
8 {
9     public static class LoggerService
10     {
11         private static readonly Lazy<ILogger> _logger;
12         private static readonly Lazy<IBatchLogger> _batchLogger;
13
14         static LoggerService()
15         {
16             var fileName = $"Log_{DateTime.Now.ToFileTime()}.txt";
17             var writer = new Lazy<IBatchWriter>(() => new WriterPool(new List<IWriter> { new ConsoleWriter(), new FileLogWriter(fileName)}));
18             _logger = new Lazy<ILogger>(() => new Logger(writer.Value));
19             _batchLogger = new Lazy<IBatchLogger>(() => new BatchLogger(writer.Value));
20         }
21
22         public static ILogger GetLogger()
23         {
24             return _logger.Value;
25         }
26
27         public static IBatchLogger GetBatchLogger()
28         {
29             return _batchLogger.Value;
30         }
31     }
32 }
```

3. Результат представить в виде решения (solution) Visual Studio 2017 (2015), упакованным в архив (или прислать ссылку на репозиторий, размещенный на GitHub/GitLab).

Наличие юнит тестов, использование шаблонов проектирования, DI контейнера и принципа инверсии зависимости будет плюсом.