**Word puzzle application document**

**Application architecture:**

Console application based on the dependency injection design pattern and decomposed into models, services and tools.

**Models:**

* MySettings -> To bind the appsettings json model.
* Argment -> To get the user input.

**Services:**

* ManagerService: The main service called by the main of the console application.
* LoadTextAndGetWordsList: To load the text from the file and put in a table of words.
* UserInputServices: To read the argument from the user and to validate the data annotation defined in the argument model.

**Tools:**

* AppStartup: To bind the appsettings and to apply the dependency injection.
* ExceptionHandling: To catch and to style the exceptions.
* FilesInputOutput: The files reader and writer.

**Application setting:**

**appsettings.json** to set the application.

* DataSource: location to import the dictionary data file,
* DictionaryFile: Input File name,
* ResultFolderName: location to export the result file,
* WordLength: The desired word length.

**How the algorithm problem it was solved:**

1. Get indexes of StartWord and EndWord.
2. Get a list of all the words between the 2 indexes with length of 4.
3. Loop the words of the list to get all possibility with difference of one letter.
4. Get the last word from the possible word and add it to the final result.

**Used NuGet packages**

* NUnit
* Microsoft.Extensions.Hosting (Configuration, DependencyInjection, options)
* Serilog.Sinks.RollingFile

**Next version future features:**

* Security

**Used References:**

* <https://www.lambdatest.com/blog/nunit-vs-xunit-vs-mstest/>