QA Training

Avanade Practical Project

Ugne Sarakauskaite

June 23, 2021

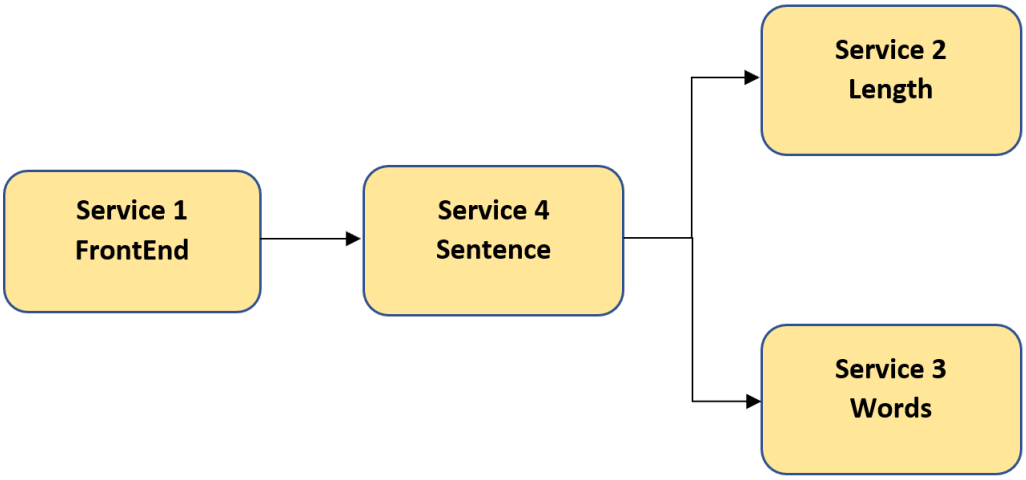
Contents

1 IDEA

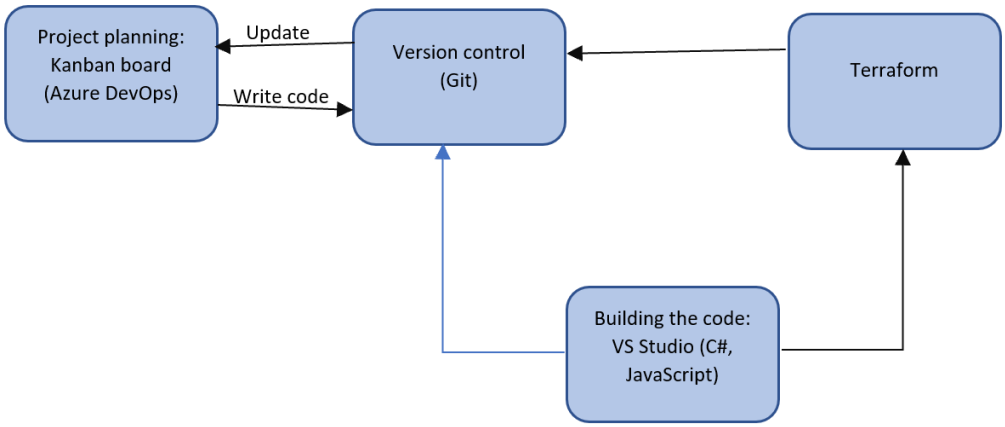
An app that implements a sentence generator. It works based on 4 services:
Service 1 is responsible for the front-end of an app;
Service 2 (== length project) randomly selects a number from 1 to 5. That number is X (say) and it will represent the length of the sentence;
Service 3 (== words project) randomly selects 5 words from CSV files (one from each: adjectives, adverbs and verbs, and two from nouns), these words are then stored in a dictionary;
Service 4 (== sentence project) merges services 2 and 3 in such a way, that the sentence of length X is generated. It is constructed following the structure below:

switch *length_X*
- case 1: Noun
- case 2: Adjective + Noun
- case 3: Adverb + Adjective + Noun
- case 4: Verb + Adverb + Adjective + Noun
- case 5: Verb + Adverb + Adjective + Noun + AND + Noun

Relation between services (ER Diagram):



Developer Diagram:



Relevant links:
Kanban board
GitHub repo

2 RISK ASSESSMENT

2.1 Table

	Description	Evaluation	Likelihood	Impact Level	Response	Control Measures
1	Functionality of the webapp does not meet the requirements	Project does not work as intended	Medium	Medium	Work on project consistently, ask for help, do more practice	Work consistently, ask for help in order to implement desired features.
2	Implemented functionality works in a wrong way	Project criteria is not met	Low	Medium	Edit code appropriately	Test application consistently using various testing techniques
3	Application is not finished due to some issue that holds from further development	Project is incomplete, deadline is not met	High	High	Debug the code, do research considering the issue online, ask for help from fellow colleagues and superiors	Coding carefully, constantly rebuilding and testing the system.
4	Secret credentials (such as information about azure database) are shared publicly	Sensitive data is exposed – increased possibility of external attack	Medium	High	Remove and change exposed data as soon as issue is spotted, report what happened	Before committing anything add sensitive files to .gitignore
5	Broken version of web app deployed onto production	Deployed product may have issues or missing requirements or may not work	Low	High	Revert production to most recent stable version	Test web app before and after its deployment
6	Design patterns and good coding practices are not applied	Code is hard to read, reuse or maintain	Low	Medium	Perform robust refactoring of the code	Refactor/test code consistently, follow basic coding standards and use at least MVC pattern
7	Would not be able to deploy the project	Services only works locally	Medium	Medium	Explain possible reasons behind it, try to find solution even after the deadline	Do research, debug, ask for help

2.2 Revisits

8. *2021/06/16*

Likelihood: High.

Notes: Having trouble deploying webapps which use paths (for reading from text files). Trying to do research and ask for help. Tried to add resources path to appsetting.json, did not work

2021/06/18:

Likelihood: VERY high

Notes: unsuccessfully attempting to deploy the services for a week. My first guess was that code involving paths (specifying from where to read resources) is not implemented correctly or reading files from local repository may be an issue while deploying an application. I hence created an azure storage account and tried to read files from blobs. Even so, deployment was still unsuccessful.

2021/06/23

Likelihood: Low

Notes: I have successfully deployed the project. The problem was simply configuration of application settings for web apps.

1. *2021/06/22* **Likelihood**: Low
4. *2021/06/22* **Likelihood**: Low

Notes: While working on the first project I have learned how to work with .gitignore file and also how to recursively eliminate commits. During this project I have learned how to work with sensitive data in Terraform (**4.1**) and also how to safely configure azure credentials in yaml file (**5**)
7. *2021/06/22* **Likelihood**: Low

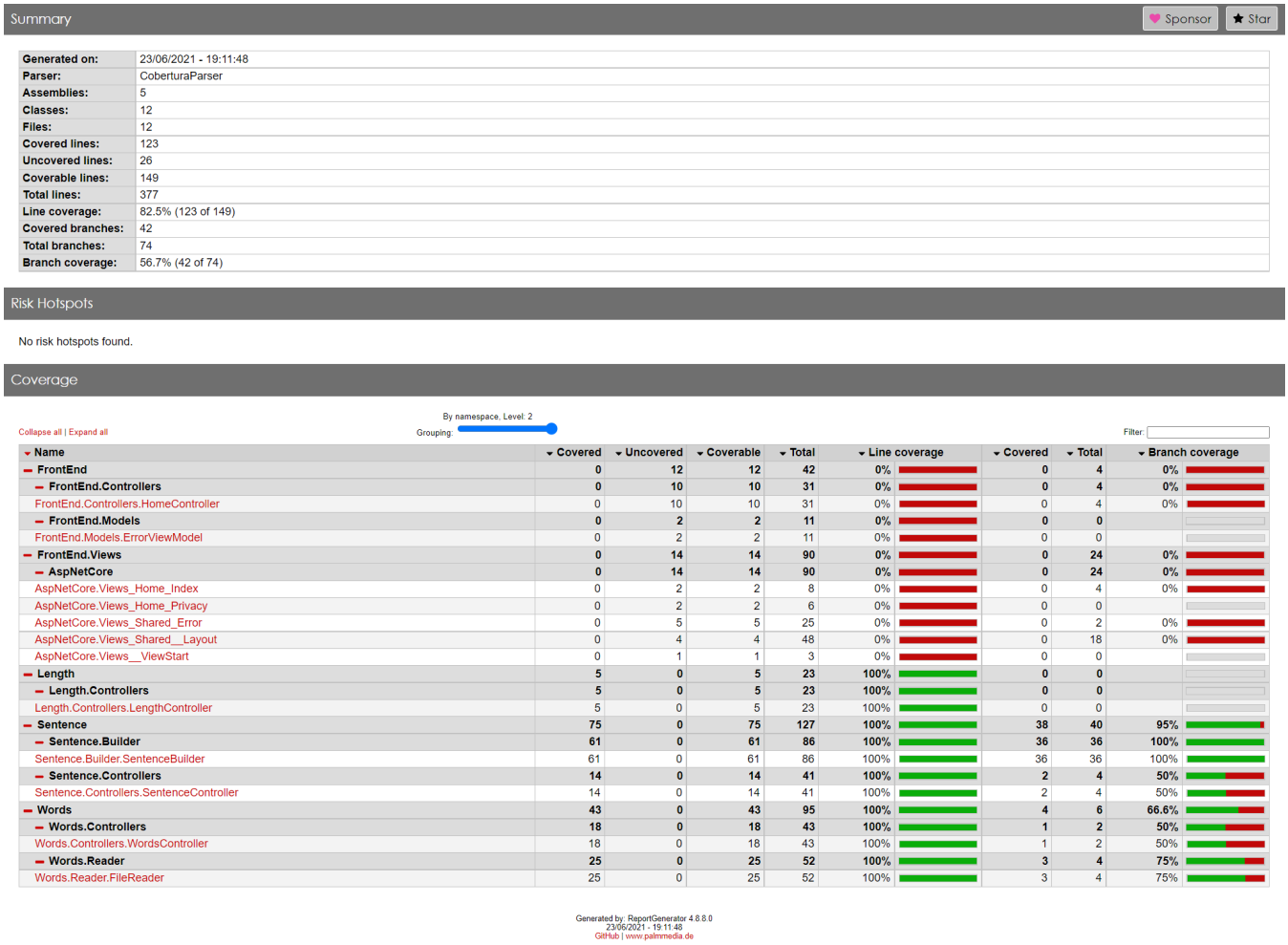
Notes: project is now compliant with the requirements

3 TEST DRIVEN DEVELOPMENT

While developing each service I attempted to use a well-known testing technique: test driven development. Hence, before creating each class, I used to think what that class is intended to do and write a test for it. As such, I have tried to write a unit test to check a single functionality. After a 'failing test' is written, I used to write just enough code to satisfy condition in test.

By following this technique, I was able to develop the code more efficiently and avoiding unnecessary errors. Also, TDD allowed me to visualise 'how' the program should function, hence faster obtain the desired results.

Code Test Coverage:



Summary

Sponsor ★ Star

Generated on:	23/06/2021 - 19:11:48
Parser:	CoberturaParser
Assemblies:	5
Classes:	12
Files:	12
Covered lines:	123
Uncovered lines:	26
Coverable lines:	149
Total lines:	377
Line coverage:	82.5% (123 of 149)
Covered branches:	42
Total branches:	74
Branch coverage:	56.7% (42 of 74)

Risk Hotspots

No risk hotspots found.

Coverage

Collapse all | Expand all

By namespace, Level: 2
Grouping:

Filter:

Name	Covered	Uncovered	Coverable	Total	Line coverage	Covered	Total	Branch coverage
FrontEnd	0	12	12	42	0%	0	4	0%
FrontEnd.Controllers	0	10	10	31	0%	0	4	0%
FrontEnd.Controllers.HomeController	0	10	10	31	0%	0	4	0%
FrontEnd.Models	0	2	2	11	0%	0	0	
FrontEnd.Models.ErrorViewModel	0	2	2	11	0%	0	0	
FrontEnd.Views	0	14	14	90	0%	0	24	0%
AspNetCore	0	14	14	90	0%	0	24	0%
AspNetCore.Views_Home_Index	0	2	2	8	0%	0	4	0%
AspNetCore.Views_Home_Privacy	0	2	2	6	0%	0	0	
AspNetCore.Views_Shared_Error	0	5	5	25	0%	0	2	0%
AspNetCore.Views_Shared__Layout	0	4	4	48	0%	0	18	0%
AspNetCore.Views__ViewStart	0	1	1	3	0%	0	0	
Length	5	0	5	23	100%	0	0	
Length.Controllers	5	0	5	23	100%	0	0	
Length.Controllers.LengthController	5	0	5	23	100%	0	0	
Sentence	75	0	75	127	100%	38	40	95%
Sentence.Builder	61	0	61	86	100%	36	36	100%
Sentence.Builder.SentenceBuilder	61	0	61	86	100%	36	36	100%
Sentence.Controllers	14	0	14	41	100%	2	4	50%
Sentence.Controllers.SentenceController	14	0	14	41	100%	2	4	50%
Words	43	0	43	95	100%	4	6	66.6%
Words.Controllers	18	0	18	43	100%	1	2	50%
Words.Controllers.WordsController	18	0	18	43	100%	1	2	50%
Words.Reader	25	0	25	52	100%	3	4	75%
Words.Reader.FileReader	25	0	25	52	100%	3	4	75%

Generated by: ReportGenerator 4.8.8.0
23/06/2021 - 19:11:48
GitHub | www.palmmedia.de

3.1 Tests implemented

LengthControllerTest: This is the simplest service that has been developed, as such tests here work in a very straightforward way. First, it is checked if produced result is of type integer, and if that integer is in scope from one to five (to confirm that service is called in loop for a thousand times). Lastly, tests verify that minimum value of one (and maximum of five) is obtained by calling service until the desired number occurs.

FileReaderTest: tests verifies that *ReadBlobContents* method produces non-null result, confirms the existence of all, Azure Storage Account and container named project and also the blobs. Moreover, it tests that result obtained by *ReadFile* method is also not null and that its obtained array of strings contains (randomly chosen) words from the blob. Lastly, tests check that string list built by *GetRandomWords* method has any elements as well as verifies number of elements randomly selected from each blob (1 element each, and two from nouns blob).

WordsControllerTest: Here tests check that words obtained after the service is called contains any elements. Then, as words are stored into a dictionary, it is verified that four keys, for adjectives,

adverbs, verbs and nouns, are stored. Lastly, it is confirmed that dictionary had five non-empty values overall (one for each speech part and two for nouns).

SentenceBuilderTest: Test for *SentenceBuilder* class has pre-defined *wordsDummy* string which represents example value of words dictionary obtained by words service. Data retained by calling words service in *SentenceController* is given as a string output, hence tests check that *StoreWords* method re-stores words back to dictionary data structure. As so, it verifies if four keys, namely adjectives, adverbs, verbs and nouns, are stored with one non-empty value each, and two values for nouns key. *SentenceBuilderTest* also verifies that given the length and words values, the BuildSentence constructs output in a correct structure (described in IDEA section). Note: after testing SentenceController, I have learned that best practice to check SentenceBuilder class would also be to utilise mock object, nonetheless, this will be noted as a future goal for further developments.

SentenceControllerTest: *SentenceController* is tested using mock object. The process is described below.

3.2 Mocking

To test *SentenceController*, which in order to build a sentence merges length and words service, I have utilised *mock object* for mimicking http request calls. Mock object first is used to set the results of length and words service and then it is utilized as a handler to create a new session for http request to be sent to sentence service (=> to create new *httpClient* object). Finally, after sentence controller is initiated with mock *httpClient*, sentence service is called, and it is confirmed that controller produced the result expected.

3.3 Issues I have faced

One of the services that I have implemented (*Words* project) was initially retaining information from four text files populated in *resources* folder. Thus, to read the text files, I had to find a way to specify the path of the folder. My first guess was to utilise *Directory.GetCurrentDirectory* method in *FileReader* class, which then would be called in *WordsController*. As a result, webapp passes integration testing, however, the unit tests fail. That is because once testing class tries to obtain the current path (in *FileReader* class), it obtains its own directory (as should) and hence cannot find the resources folder. After some research, I came up with the solution to duplicate the folder in two projects and utilise *Directory.GetDirectoryName* in combination with Regex class to eliminate going further path (e.g., going to bin folder) than needed (github commit).

While trying to deploy services, I have been advised to try and read the information from *Azure Blobs* rather than text files. To do so, I had to use a connection string to connect to *Azure Storage Account* where the container with the blobs is stored. As usual, to not expose sensitive information, I have stored the connection string in *appsettings.json* file, however, I did not know how to access it in testing project. After some research, I have decided to duplicate the appsettings in *SentenceGeneratorTest* project which solved the problem. I believe there is a way to utilise mock objects for this purpose, however this would be a future goal to learn.

Note: Duplicating appsettings was also helpful while testing *SentenceController* as it allowed me to retain web addresses for length and words services.

3.4 Future improvements

I believe that I still must learn a lot about utilising mock object and explore their functionality. Personally, although I have heard about this testing technique a while ago, using mocks is still a new and very appealing methodology for me to study in programming. Thus, in future projects my aim would be to implement mocks in every test class in order to practice and hence better understanding about the subject.

Furthermore, as nature of test-driven development is, I need to improve writing more accurate tests, covering all possible outcomes of code that is being developed. This way I would avoid more errors that may occur and produce more accurate product.

4 TERRAFORM

4.1 Terraform in project

Terraform is a very helpful infrastructure as code software tool, which extremely eases azure resource management. I believe I will be widely using *Terraform* in future developments, however, I still need a lot of practice and research to do in order to utilise it to its full capacity.

For this project, I have managed to create my resource group, app service plan (together with four app services with unique app settings) and a storage account.

I have also learned how to protect sensitive information from being exposed. To do so, I have followed the tutorial, that explains how s*ecrets.tfvars* file is working and demonstrates use of *sensitive = true* tag for variables.

4.2 Issues I have faced

One of the issues that I have come across while practicing use of *Terraform* was creating web apps. I have since learned that when using an App Service Plan in the Free (or Shared) tiers, a $use\_32\_bit\_worker\_process$ must be set to *true* in $site\_config$ block (ref).

Another problem that I have come across was pushing *Terraform* (that was initialised, planned and applied) onto Git. When *Terraform* code is initiated, large configuration files are automatically created, hence they cannot be pushed onto remote control. I realised that I have not added additional Terraform-related bit to *.gitignore* file, which would avoid making unnecessary pushes. To add correct files to be ignored I used this git repository as a reference. However, as I have already attempted an invalid commit, I had to reverse git history to one before these commits. To achieve that, I used command $git\ reverse\ --hard\ @\{u\}$, which reversed my code to the last push I made, i.e., to remote version of the project in current branch.

However, I did not manage to use *Terraform* to create azure storage containers. As was outlined in the error message, this is due to the non-public access level of the container, nonetheless, even after manually enabling public access on azure portal, same error is still present.

5 DEPLOYMENT

In order to comply with CI/CD I had to implement *Pipelines*. This way, webapp is redeployed automatically each time after the functionality of the app is changed and pushed onto GitHub main branch. For this project, I have used *GitHub Actions* to deploy my application.

While trying to deploy the project I have faced quite a few issues. One of them was configuration of GitHub secrets. After attempting to publish a webapp, I was given a 'GitHub secret does not exist' error. Hence, after some time looking into a problem, I have found a guide to manually add a secret for webapp. Following this tutorial, I learned how to obtain both, azure credentials and publish profile of an app service.

6 GIT

6.1 Git in project

To develop, track and manage changes in code for this project I have used Git version control. Throughout the development of the four services, I have created branches for each feature (class) that I have implemented, made edits to the code and finally merged feature branch with main one. This way I was able to retain all previous information I needed.

Moreover, to be as accurate as possible, in each commit message I used to mention what I have changed and how that change has impacted the services. This then made it easier to find information in my git logs.

6.2 Future improvements on version control

As I have planned my project using *DevOps* kanban board, I recorded my epics, user stories and tasks there. This time, I tended to create a branch for each user story rather than a task. As such, I aim to practice better use of Git and create a branch for each task that is set to be done.

7 FUTURE IMPROVEMENTS FOR THE PROJECT

Software engineering is a never ending process of development. Nonetheless, sometimes it is better to follow the principle of minimum value product (MVP) in order to meet all the necessary requirements. Thus,if not limited by the constraint of time, I would have tried to get my hands on Docker containers.

One of the reasons to use Docker for this project is simply because it complies perfectly with the requirements. It would decouple the four services and the infrastructure which would not only result in more lightweight app components but also allow faster delivery of the software. Furthermore, it would be a beneficial and also interesting tool to explore and practice as Docker is widely used in many projects and would potentially help with my future career.