Ap18B Modul 122 | Eggenberger Timon

Dokumentation

Inhalt

[Einleitung: 3](#_Toc67002907)

[Entwicklungsumgebung: 3](#_Toc67002908)

[Ablauf des Scripts 4](#_Toc67002909)

[Fazit: 4](#_Toc67002910)

[Quellen 5](#_Toc67002911)

# Einleitung:

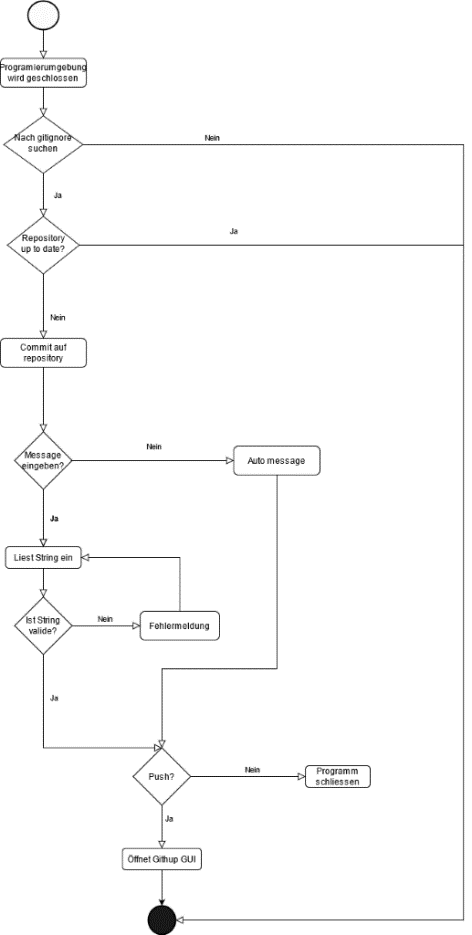
In diesem Projekt sollten wir ein Script entwerfen, welches über einen Entrypoint selbstständig Prozesse ausführt. Um ein Relativ praxisnahes und für mich nützliches Script zu machen habe ich mich für eines Entschieden das automatisch Projekt Daten auf git pusht. Als Informatiker kennen wir alle sicher die Situation: man hat vergessen zu pushen und jetzt ist der ganze fortschritt weg oder wie bei mir, der Laptop geht kaputt. Nun zu dem wird es jetzt nicht mehr kommen dank dieses Scripts. Dies war der Grundgedanke da ich selber oft vergesse am ende des Tages meinen Fortschritt zu pushen.

# Entwicklungsumgebung:

Als Entwicklungsumgebung habe ich mich für Visual Studio code entschieden, weil ich selbst im Geschäft oft damit arbeite und mir die Umgebung daher sehr vertraut ist. Auch sorgt dies dafür das, wenn ich das Projekt in VS code umsetze ich auch praktisch davon profitieren kann. Auch JavaScript als Sprache ist mir mehr als vertraut.

Online habe ich mich etwas mit der Umsetzung auseinandergesetzt und dabei ein großartiges Video gefunden was mich auf die Idee brachte das Skript als VS code Extension umzusetzen was mir vorher gar nicht in den Sinn kam. Auch wurde in diesem toll Erklärt, wie eine Solche Extension veröffentlicht werden kann.

# Ablauf des Scripts:



Beim Ablauf wollte ich mich an unsere zuvor festgelegten Ziele halten. Hierbei traten aber ein Paar Probleme auf. Als erstes wollte ich den Entrypoint beim Öffnen von VS Code setzten und dann in einer while true schleife laufen. Dies war aber sehr ineffizient und funktionierte nicht richtig. Deshalb musste ich den Entrypoint so setzten das er aktiviert wird sobald Visual Studio Code geschlossen wird. Auch musste ich einen Workaround findend da ich auch nach langem Suchen keine Github library für JS gefunden habe. Das Ganze konnte aber mit der exec Methode ganz leicht umgangen werden.

Aus logistischen gründen habe ich mich auch für die CWD Methode anstatt der dirname Methode entschieden da dirname einfach die Directory des Source codes zurückgibt

Das grösste Problem allerdings war, das ich für die Eingabe eines eigenen Kommentares vor dem Pushen, ein CMD öffnen wollte. Dies ist aber unter Windows 10 aus Sicherheitsgründen nicht möglich. Ich habe mich also dafür entschieden einfach einen Kommentar mithilfe von new Date() autogenerieren lassen. So kann zwar nicht Selber entschieden werden was für eine Message gepuscht wird aber man kann anhand vom Zeitstempel ablesen wann das ganze Gepuscht wurde.

Sobald nun VS Code geschlossen wird, prüft das Script ob sich in der zuletzt geöffneten Directory ein .gitignore File befindet. Danach wird geschaut, ob das Repository auf dem neusten Stand (up to date) ist. Sollte dem nicht so ein wird zuerst ein commit mit dem momentanen Zeitstempel generiert und danach das ganze auf GitHub gepusht.

# Fazit:

Ich hate Spass an der Umsetzung dieses Projektes. Es stellte einen meiner ersten Berührungspunkte mit Script Anwendungen dar und war deswegen etwas Neues für mich. Die Umsetzung ist mir eher so la la gelungen. Gerne hätte ich noch die Funktion eingebaut einen eigenen Kommentar einzugeben oder auch ein Autoversionierungsstyem. Ich fand es toll wie Simpel es war die Extension zu Veröffentlichen und ich kann mir gut vorstellen dieses Feature in der Zukunft auch öfters zu nutzten in Verbindung mit Scripts. Nach wie vor finde ich auch erstaunlich wie kurz mein Script effektiv ist.

Quellen:   
informatives YouTube Video: <https://www.youtube.com/watch?v=WQ0Mf-5noRc>  
  
Exec: <https://stackoverflow.com/questions/27753246/match-vs-exec-in-javascript>

<https://www.w3schools.com/jsref/jsref_regexp_exec.asp>

Dirname vs CWD:

<https://stackoverflow.com/questions/9874382/whats-the-difference-between-process-cwd-vs-dirname>