**Goprojekt\_lukasnachtigall\_18112022**

**API-Integration in golang:**

**Bereits erledigt (Stand: 18.11.2022):**  
Die Aufgabe habe ich begonnen indem ich mir mein gesamtes webprojekt noch einmal angeschaut habe, also sowohl den HTML, als auch den CSS und Javascriptcode und insbesondere die API Integration mit node.js ([solyvista/to-do-website: This is still a work in progress. The goal of this project is to build a website with an integrated to-do-list that can be individually adjusted to the users needs. The website is written in HTML, JavaScript and CSS with bootstrapintegration and an integrated jQuery API connection. (github.com)](https://github.com/solyvista/to-do-website)).  
Danach habe ich bei Google, GitHub, Youtube und LinkedInLearning unter anderem nach „golang Backend API Integration“ und „golang API“ und „golang backend“ gesucht.  
Auf Youtube habe ich ein gutes Video gefunden welches die Codestruktur für eine API Backend Integration in Go erledigt ([Consume RESTful API Endpoints Within A Golang Application - YouTube](https://www.youtube.com/watch?v=bYZlQMWR8HM)).  
Weiterhin habe ich auf GitHub ein hilfreiches repository gefunden welches eine REST API in golang an eine to-do-webseite hinzufügt ([geraldoahnert/rest-api-golang: REST API Go Lang, todo app. (github.com)](https://github.com/geraldoahnert/rest-api-golang))  
  
Quellen:  
[solyvista/to-do-website: This is still a work in progress. The goal of this project is to build a website with an integrated to-do-list that can be individually adjusted to the users needs. The website is written in HTML, JavaScript and CSS with bootstrapintegration and an integrated jQuery API connection. (github.com)](https://github.com/solyvista/to-do-website)  
[Consume RESTful API Endpoints Within A Golang Application - YouTube](https://www.youtube.com/watch?v=bYZlQMWR8HM)  
[geraldoahnert/rest-api-golang: REST API Go Lang, todo app. (github.com)](https://github.com/geraldoahnert/rest-api-golang)

**Aufgetretene Herausforderungen**Schwierigkeiten hatte ich bei der Verknüpfung des gefundenen golang Code mit dem Code von meinem Webprojekt.  
Um dies zu lösen werde ich mir noch einmal das Video ([Consume RESTful API Endpoints Within A Golang Application - YouTube](https://www.youtube.com/watch?v=bYZlQMWR8HM)) anschauen.   
Dann werde ich mir bei LinkedInLearning noch einmal ein Videokurs anschauen wie man eine REST API in node.js integriert um mein Grundlagenwissen über API-Integration noch einmal aufzufrischen.  
Weiterhin werde ich die Nachhilfe nutzen um Probleme bei der Codeimplementierung zu bearbeiten.

Quellen:  
[geraldoahnert/rest-api-golang: REST API Go Lang, todo app. (github.com)](https://github.com/geraldoahnert/rest-api-golang)  
[Consume RESTful API Endpoints Within A Golang Application - YouTube](https://www.youtube.com/watch?v=bYZlQMWR8HM)

**Noch zu tun:**  
Es muss noch die gefundene Vorlage auf mein webprojekt angepasst werden und mit der zur Verfügung gestellten API von Techstarter verbunden werden um request zu empfangen.  
Dafür muss ich die in meinem in node.js vorhanden code in go abändern.   
Wesentlich dafür sind die Erstellung einer Datei in golang geschrieben welche ein Äquivalent zu der app.js Datei meines Webprojektes bietet (diese Datei ist für die API-Integration in meinem Webprojekt zuständig). Enthalten sein müssen in der golang-Datei für die API-Integration die folgenden Funktionen: get, add, post, delete.

Quellen:  
[geraldoahnert/rest-api-golang: REST API Go Lang, todo app. (github.com)](https://github.com/geraldoahnert/rest-api-golang)  
[Consume RESTful API Endpoints Within A Golang Application - YouTube](https://www.youtube.com/watch?v=bYZlQMWR8HM)

**Änderung des Hauptprojekts zu cowsay am 18.11.2022**

Am Freitag den 18.11.2022 habe ich beschlossen erst einmal das cowsayprojekt weiterzumachen, da ich den Code als leichter zu schreiben empfand.  
  
Mein Code ist abrufbar über mein GitHubAccount in dem repository golangprojektnovember22:  
[solyvista/golangprojektnovember22: this is a small golang project. i wrote a cowsay (with a pinguin instead of a cow). the program prints a function that "draws" a pinguin. one can scan input from the terminal via a scan function. the size of the speaking bubble of the pinguin is depended on the lenght of the input from the scanner. (github.com)](https://github.com/solyvista/golangprojektnovember22)  
  
**Planung:**  
Arbeitsschritt1: Recherche betreiben, Vorlagen/Quellen finden. Hier habe ich sehr viele gute Vorlagen auf GitHub gefunden welche auch verständlich waren. Viele der Vorlagen waren jedoch äußerst lang und hatten unnötige Codeelemente.

Quellen:  
- [flaviocopes/gololcat: A lolcat implementation in Go https://flaviocopes.com/go-tutorial-lolcat/ (github.com)](flaviocopes/gololcat:%20A%20lolcat%20implementation%20in%20Go%20https://flaviocopes.com/go-tutorial-lolcat/%20(github.com))  
- [flaviocopes/gocowsay: Clone of the `cowsay` command line application - - https://flaviocopes.com/go-tutorial-cowsay/ (github.com)](flaviocopes/gocowsay:%20Clone%20of%20the%20%60cowsay%60%20command%20line%20application%20-%20-%20https://flaviocopes.com/go-tutorial-cowsay/%20(github.com))  
- [AnonymousAAArdvark/cowsay-go: Cowsay implementation written in Go (github.com)](https://github.com/AnonymousAAArdvark/cowsay-go)  
- [syohex/gowsay: cowsay implementation in Golang (github.com)](https://github.com/syohex/gowsay)  
- [dhruvbird/go-cowsay: Basic cowsay in GO! (github.com)](https://github.com/dhruvbird/go-cowsay)  
- [ShumilaMalik/Cowsay-in-Golang (github.com)](https://github.com/ShumilaMalik/Cowsay-in-Golang)  
- <https://gobyexample.com/command-line-arguments> <https://yourbasic.org/golang/gotcha-index-out-of-range/>

Arbeitsschritt2: Projektordner anlegen, Dateien in VSCode anlegen und in den Projektordner ablegen:

Arbeitsschritt3: Code schreiben.  
Um den Code zu schreiben waren folgende Arbeitsschritte erforderlich:  
In einem ersten Schritt kann die Figur (in meinem Fall ein Pinguin) einfach über die Eingabe von Zeichen „gezeichnet“ werden, die Figur muss innerhalb einer Funktion stehen und kann dann als Variable deklariert werden und im nächsten Schritt mittels print.Ln ausgegeben werden im Terminal wenn der Code abgerufen wird.  
Als zweiter Schritt muss eine Funktion geschrieben werden um den Input im Terminal einzulesen, hierfür gibt es verschiedene Möglichkeiten. Die einfachste Möglihkeit hier ist „scan“.  
Im letzten Schritt muss eine Sprechblase erstellt werden welche die Größe verändert je nachdem wie lang der Inputtext ist welcher mittels der Scanfunktion aus dem Terminal ausgelesen wird. Hierfür bietet sich eine forschleife an welche als Bedingung die Größe des Inputs aus der Scanfunktion nimmt und je nachdem ob der Input größer oder kleiner als x (beliebige Zahl definieren) einen anderen Output gibt (Output ist die Sprechblase welche groß oder klein sein kann je nach Scannerinput).  
Am 23.11 ist im Dialog über das Projekt mit einem teaching-assistant klar geworden, dass eine scan-Funktion nicht die Aufgabenstellung erfüllt. Daher habe ich die scan-Funktion ersetzt durch die os.Args-Funktion, da in der Aufgabenstellung eine Übergabe des Inputs als Argument gefordert wurde. Die os.Args Funktion erhält die os.Args-Funktion, die print.Ln-Funktion mit der Ausgabe des eingegebenen Arguments aus dem Terminal und eine for-Schleife um einen indexoutofbounderror abzufangen wenn keine Eingabe im Terminal erfolgt und somit kein Argument über os.Args ausgelesen wird. Das Abfangen einer indexoutofbounderrormeldung wurde abgefangen mit einem if-statement welches für den Fall, dass nur ein Argument übergeben wird (es wird immer mindestens ein Argument übergeben, nämlich das Argument des Dateinamens) eine errormeldung ausgegeben wird.

Arbeitsschritt4: Code testen: überprüfen ob Code läuft wie gewünscht, ggf. Verbesserungen vornehmen:  
Stand 25.11.2022: CODE FUNKTIONIERT!

Arbeitsschritt5: Dokumentation des Projektverlaufs inklusive Quellenverzeichnis fertigstellen und abgeben