
Modul 122

Leistungsbeurteilung LB2 - Dokumentation

Modul	IET-122 - Projektarbeit
Eingereicht von	David Meer, Levyn Schneider Stammgruppe/ <u>Partnerarbeit</u>
Eingereicht bei	Frau Yilmaz
Datum	18. Juni 2024

Änderungsverzeichnis

Datum	Version	Änderung	Autor
17.04.2023	0.5	Vorlage	Urs Dummermuth
28.05.2024	1.0	Dokumentation	Levyn Schneider, David Meer

Inhaltsverzeichnis

1	Ziele und Anforderungen	3
1.1	Einleitung	3
1.2	Zweck des Skriptes	3
1.3	Ziele	3
1.4	Anforderungen	3
2	Ablaufdiagramm	5
3	Skript/Programm (Realisierung)	6
3.1	Technologie Bash / Powershell / Python	6
3.2	Ein- und Ausgabe	6
3.3	Kontrollstrukturen	8
3.4	Regex	8
3.5	Datenbank	9
4	Integration	10
4.1	Implementierung	10
4.2	Sicherheit	10
4.3	Kompatibilität	10
4.4	Betrieb und Wartung	10
5	Usecases und Testfälle	11
5.1	Usecase	11
5.2	Testfall	11
6	Präsentation, Dokumentation	12
6.1	Demo-Video	12
6.2	Vortrag	12
7	Reflexion	13
7.1	Journal	13
7.2	Auswertung	14
7.3	Fazit	14
8	Anhang	15
8.1	Skript/Programm-Listing	15

1 Ziele und Anforderungen

1.1 Einleitung

Dasselbe nochmal? Nicht ganz, unser jetziges Projekt hat dasselbe Prinzip wie das vergangene, aber dieses Mal besteht das Skript aus Python-Code. Levyn kennt sich mit Python bereits sehr gut aus, da er privat wie auch im Basis Lehrjahr sehr viel zu tun damit hatte. Anders als Levyn hatte David beruflich noch nichts damit zu tun. Dafür habe ich mich in meiner Oberschulzeit damit sehr befasst, und kenne so die Programmiersprache auch grob.

1.2 Zweck des Skriptes

Wir erstellen eine CLI-E-Mail-Applikation, um einfach E-Mails im Terminal zu senden und empfangen. Dies haben wir uns ausfolgendem Grund überlegt; Falls einmal die E-Mail-Applikation nicht funktioniert, man aber trotzdem etwas empfangen oder auch senden möchte kann man dieses Tool verwenden.

1.3 Ziele

1. Bis am 25.06.2024, muss man über das CLI-Tool E-Mails versenden können.
2. Bis am 25.06.2024, muss man über das CLI-Tool ungelesene E-Mails empfangen können.
3. Bis am 25.06.2024, muss es möglich sein, ungelesene E-Mail in einem .txt speichern

1.4 Anforderungen

1.4.1 Allgemeine Anforderungen

- 1a: Es muss eine Sender-E-Mail-Adresse konfiguriert sein
- 1b: Es muss einen Gmail App Code konfiguriert sein.
- 1c: Es muss eine Empfänger-E-Mail-Adresse eingegeben werden.
- 1d: Es muss ein Betreff eingegeben werden.
- 1e: Es muss eine Nachricht eingegeben werden.

- 2a: Es muss eine Sender-E-Mail-Adresse konfiguriert sein.
- 2b: Es muss einen Gmail App Code konfiguriert sein.

1.4.2 Eingaben und Ausgaben

Eingaben:

- Absender-E-Mail-Adresse
- Gmail App-Code
- Passwort
- Empfänger-E-Mail-Adresse
- Betreff
- Nachricht
- Menu-Auswahl

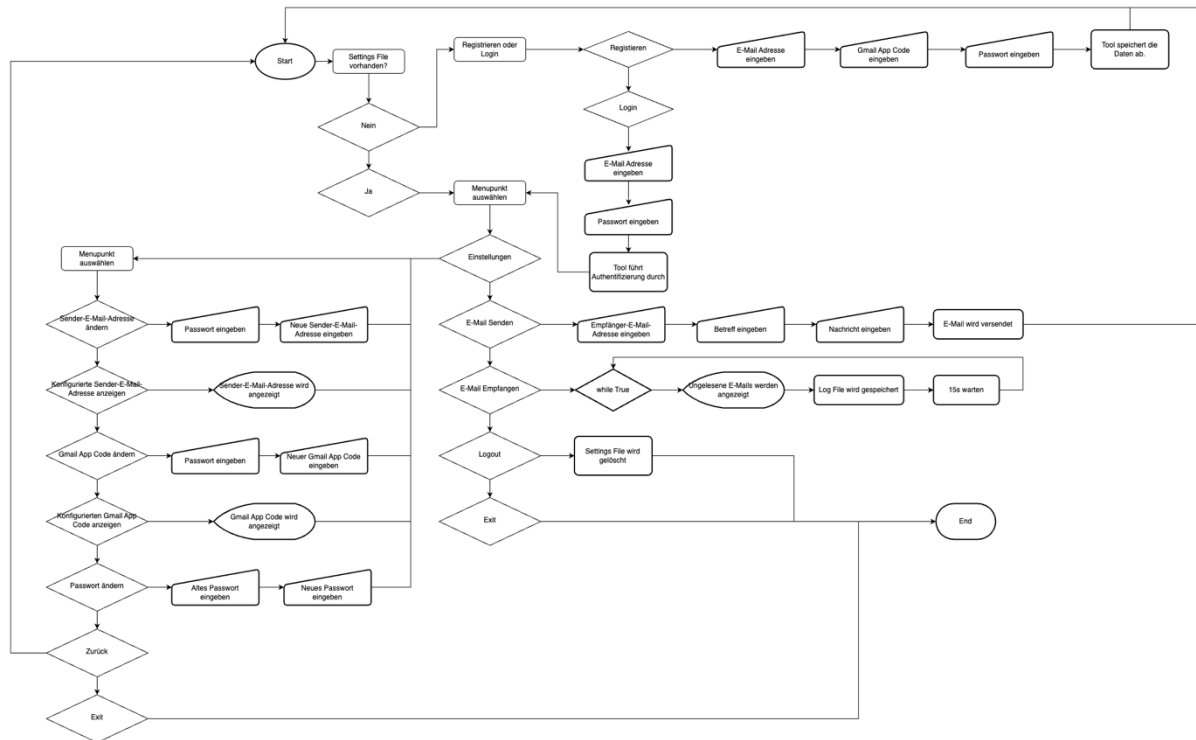
Ausgaben:

- Konfigurierte Sender-E-Mail-Adresse
- Konfigurierten Gmail App-Code
- Ungelesene E-Mails
- .txt Log-Dateien von den Empfangenen E-Mail.

1.4.3 Programmtechnische Anforderungen

- Die Python Version muss über, oder gleich wie, Version 3.10 sein.
- Der Nutzer muss eine Gmail E-Mail besitzen.
- Der Nutzer muss eine Gmail App-Code erstellt haben.

2 Ablaufdiagramm



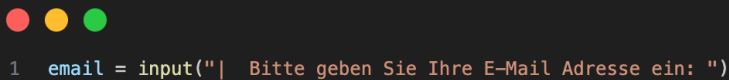
3 Skript/Programm (Realisierung)

3.1 Technologie Bash / Powershell / Python

Für dieses Projekt nutzten wir Python Version 3.10.

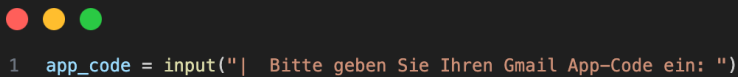
3.2 Ein- und Ausgabe

Eingabe für die eigene E-Mail-Adresse:



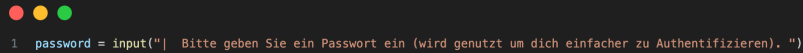
```
1 email = input("| Bitte geben Sie Ihre E-Mail Adresse ein: ")
```

Eingabe für den Gmail App-Code:



```
1 app_code = input("| Bitte geben Sie Ihren Gmail App-Code ein: ")
```

Eingabe für das Passwort:



```
1 password = input("| Bitte geben Sie ein Passwort ein (wird genutzt um dich einfacher zu Authentifizieren). ")
```

Eingabe für die Menu-Auswahl:

```
1 print("Bitte wählen Sie eine Option:")
2 print("1 (Einstellungen)")
3 print("2 (E-Mail Senden)")
4 print("3 (E-Mail Empfangen)")
5 print("4 (Logout)")
6 print("5 (Exit)")
7 print()
8
9 option = input("Option wählen: ")
10
11 if option == "1":
12     os.system("python3 settings.py")
13 elif option == "2":
14     os.system("python3 send.py")
15 elif option == "3":
16     os.system("python3 receive.py")
17 elif option == "4":
18     os.remove("settings.txt")
19     print("Erfolgreich ausgeloggt")
20 elif option == "5":
21     print()
22     print("Auf Wiedersehen!")
23 else:
24     print("Ungültige Option")
```

Eingabe für die Empfänger-E-Mail-Adresse für das Versenden einer E-Mail:

```
1 to_email = input("Gebe die E-Mail Adresse ein, an die die E-Mail gesendet werden soll: ")
```

Eingabe für den Betreff einer E-Mail:

```
1 subject = input("Gebe den Betreff der E-Mail ein: ")
```

Eingabe für die Nachricht einer E-Mail:

```
1 body = ""
2 while True:
3     line = input("Gebe den Inhalt der E-Mail ein (oder 'SEND' um zu senden der E-Mail): ")
4     if line == "SEND":
5         break
6     body += line + "\n"
```


3.3 Kontrollstrukturen

Kontrollstruktur zur Funktion der Authentifikation mit E-Mail-Adresse und Passwort:

```

1  email = input("| Bitte geben Sie Ihre E-Mail Adresse ein: ")
2
3  cur.execute("SELECT password, app_code FROM account WHERE email = ?", (email,))
4  row = cur.fetchone()
5
6  if row:
7      password = None
8
9      while password is None:
10         password = input("| Bitte geben Sie Ihr Passwort ein: ")
11
12         stored_hashed_password = row[0]
13
14         # Verify the input password against the stored hashed password
15         if bcrypt.checkpw(
16             password.encode("utf-8"), stored_hashed_password.encode("utf-8")
17         ):
18             print("|")
19             print("| Login erfolgreich")
20             print("|")
21             print(
22                 "-----"
23             )
24             print()
25             with open("settings.txt", "w") as f:
26                 f.write(f"{email}\n{row[1]}")
27         else:
28             print("|")
29             print("| Das Passwort ist inkorrekt")
30             password = None
31     else:
32         print("|")
33         print("| Es existiert kein Account mit dieser E-Mail Adresse")

```

3.4 Regex

Regex Überprüfung für die Validierung einer korrekten E-Mail-Adresse:

```

1  if re.match(r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$", email) and email.endswith("@gmail.com"):

```

Regex Überprüfung für die Validierung der Anforderungen eines Passworts:

```

1  if not re.match(r"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$", password):

```

3.5 Datenbank

Wir benutzen SQLite Version 2.6.0 als Datenbank in Python.

4 Integration

4.1 Implementierung

4.1.1 Gmail E-Mail-Adresse

Das Script unterstützt nur Gmail Adressen. Erstelle deshalb auf gmail.com ein Konto oder benutze ein bestehendes Google Konto.

4.1.2 Gmail App Code

Das Script basiert, zu der Gmail Adresse, noch auf einen Gmail App Code. Erstelle einen Gmail App Code über <https://myaccount.google.com/apppasswords> oder nutze einen bestehenden.

4.1.3 Dateien

Füge alle Dateien in einen gemeinsamen Ordner.

4.1.4 Ausführung

Führe die Datei `script.py` in deinem Ordner aus und folge den Anweisungen.

4.2 Sicherheit

Um Optimale Sicherheit zu gewähren, empfehlen wir täglich oder wöchentlich ein Backup der accounts.db Datenbank zu erstellen, damit die Accounts nicht verloren gehen. Dazu wird bei jeder Registration eines Accounts das Passwort gehashed. Dies bedeutet, dass das Passwort nicht einzusehen ist.

4.3 Kompatibilität

Das Script ist auf allen Geräten Kompatibel welche Python Version 3.10 unterstützen.

4.4 Betrieb und Wartung

Nach dem Erhalt der Applikation sind Sie selbst für den Betrieb und Wartung der Applikation zuständig. Neue Features können angefragt und neu implementiert werden.

5 Usecases und Testfälle

5.1 Usecase

Unser Tool kann gut als privat Person genutzt werden, um E-Mails zu schreiben, wenn der E-Mail-Client von Gmail nicht funktioniert. Das ganze Skript könnte man aber auch noch über mehrere Anbieter erweitern (iCloud, GMX, Hotmail, etc.)

5.2 Testfall

NR	Testfall	Erwartetes Resultat	Resultat	Getestet von
01	Das Programm sucht nach dem `settings.txt` File und findet es nicht.	Das Programm erstellt ein File und fragt den Nutzer nach E-Mail und Gmail App Code.	Erfolg	Levyn
02	Das Programm sucht nach dem `settings.txt` File und prüft auf die Korrektheit.	Das Programm fährt normal fort.	Erfolg	Levyn
03	Beim Menu-Selektor gibt der Nutzer eine ungültige Eingabe.	Das Programm stoppt.	Erfolg	Levyn
04	Das Programm fragt nach einer E-Mail-Adresse. Der Nutzer gibt eine nicht valide Adresse ein.	Das Programm wiederholt die Aufforderung, solange bis die E-Mail-Adresse valide ist.	Erfolg	Levyn

6 Präsentation, Dokumentation

6.1 Demo-Video

-

6.2 Vortrag

-

7 Reflexion

7.1 Journal

Tätigkeit	Person
PowerShell-CLI-Tool nochmals angeschaut und überlegt, wie wir das ganze abändern können.	David und Levyn
Nach und nach das Skript von PowerShell in Python konvertiert	David und Levyn
Nach der Konvertierung, Skript getestet	David und Levyn
SQLite Datenbank implementiert	David und Levyn
Das Skript aktualisiert mit Login und Registration mithilfe der neu implementierten Datenbank	David und Levyn
Neue Regex-Matching-Patterns für Passwort Bedingungen implementiert	Levyn
Fertigstellung und Testing des Skripts	Levyn und David
Nach der Konvertierung das Skript getestet	David und Levyn

7.2 Auswertung

Das Projekt ist an sich wieder sehr gut gelaufen. Das Abändern von PowerShell zu Python machte uns nicht viel Arbeit. Durch die Dokumentation des letzten Projektes fiel auch das Schreiben dieser Dokumentation um einiges einfacher.

7.3 Fazit

7.3.1 David

Das Projekt war ein voller Erfolg. Mit Levyn zusammen habe ich einiges über Python dazugelernt. Insgesamt habe ich mir das ganze schwieriger vorgestellt, da wir das ganze aber schon in PowerShell gemacht haben, war es nicht so schwierig. Ebenso war unsere Zusammenarbeit sehr gut. Falls ich einmal etwas nicht begriff, hat mir Levyn es so erklärt, sodass ich einen guten Lehreffekt hatte und das Ganze dann problemlos selbst erledigen konnte oder mindestens verstand. Wir waren uns meist einig und hatten keine grossen Komplikationen. Weitere Projekte kann ich mir zukünftig gut mit Levyn vorstellen, nur verlässt er leider unsere Klasse.

7.3.2 Levyn

Ich fand das Projekt spannend den PowerShell Code in Python Code zu ändern. Ich konnte einiges neues zu Python und deren E-Mail-Funktionen lernen. Die Umsetzung des Projektes ist uns gut gelungen und wir konnten ein Skript liefern, welcher auch ein wirklicher Nutzer hat. Mit David zu arbeiten war wie auch letztes Projekt angenehm. Wir hatten nie wirklichen Stress da wir wieder so viel wie möglich vorgearbeitet haben und somit auch mehr leisten konnten.

8 Anhang

8.1 Skript/Programm-Listing

Das Skript ist ungeeignet in der Dokumentation, wenn man das Skript einfach reinkopieren würde. Deswegen ist hier der GitHub Link: <https://github.com/ly-schneider/cli-email-tool/tree/python>