Containervirtualisierung eines Terminplaners mittels Docker

Schriftliche Ausarbeitung im Fach Betriebssysteme

von

Frau Laura Luise Becker

im Studiengang Elektrotechnik/Informationstechnik

an der HAWK Hochschule für angewandte Wissenschaft und Kunst
Hildesheim / Holzminden / Göttingen
Fakultät Ingenieurwissenschaften und Gesundheit in Göttingen

3. August 2021

Inhaltsverzeichnis

1 Einleitung

In diesem Bericht wird beschrieben, wie ein Terminkalender in einem Container mittels der Virtualisierungsplattform Docker implementiert wird. Das Implementieren des Terminkalenders in einen Container soll die einfache Übertragbarkeit von Containern auf verschiedene Betriebssysteme zeigen.
Um möglichst schnell und einfach eine eigene Anwendung auf unterschiedlichsten Systemen laufen zu lassen und auch um auf die immer kürzer werdenden Update-Zyklen zu reagieren bietet sich eine Containerlösung an. Durch die Containervisualisierung ist man unabhängig von dem System und kann auf einer Hardware mehrere Betriebssysteme und Softwarevarianten entwickeln und betreiben.
Docker bietet eine benutzerfreundliche und sehr bekannte Plattform für die Containervisualisierung an. Es ist einfach mit Docker einen Container zu implementieren und dadurch eine Anwendung auf unterschiedlichen Betriebssystemen auszuführen. Obwohl sich im Laufe des Projekt doch einige Schwierigkeiten ergaben, was die unterschiedlichen Betriebssysteme anging.
Ein Terminkalender wurde gewählt, da damit trotz einfacher und bekannter Anwendung, viele unterschiedliche Aspekte der Containerisierung getestet werden kann. So wurde mit einer graphischen Benutzeroberfläche gearbeitet und auch ein Verzeichnis wurde angelegt zum Abspeichern der Termine. Der Kalender sollte drei Funktionen haben: das Anlegen neuer Termine, das Löschen von Terminen und die Anzeige von Terminen.
In dem Bericht wird kurz erläutert was Docker und Python (die verwendete Programmiersprache) sind. Danach wird die Implementierung der Anwendung und des Containers beschrieben. Zum Schluss wird noch ein Fazit gezogen.

2 Grundlagen der Technik

2.1 Docker

Docker ist ein Programm, das es mit Container Virtualisierung ermöglicht, Anwendungen unabhängig von der Hardware zu benutzen. Einzige Voraussetzung ist das Installieren der Docker Umgebung.
Mit einem Image wird von Docker der benötigte Container für die ausführbare Anwendung gebaut. Das Image wird durch einen Dockerfile, das die nötige Umgebung beschreibt, und der Anwendungssoftware gebaut. Dieses Image enthält das Dateisystem des Containers und muss außerdem alle benötigten Abhängigkeiten, Frameworks und Bibliotheken, welche das zugehörige Programm zum Ausführen benötigt, beinhalten. Das Image wird schichtartig aufgebaut, so kann ein anderes Image als Grundlage für das eigene benutzen werden. Das erleichtert und verkürzt das Entwerfen eines Dockerfiles.
Außerdem muss bei einer Veränderung an einer Schicht des Images nur dieser Teil neu erstellt werden. Das führt zur schnelleren Herstellung des bearbeiten Images, da lediglich dieser Teil neu heruntergeladen und installiert werden muss.
Der Container ist die ausführbare Instanz eines Images. Als Standards sind Container relative isoliert von dem System. Der Anwender kann selbst wählen, wie isoliert das Container Netzwerk, der Speicherplatz und andere Subsysteme sind. Der Container läuft nur solange, wie auch die Anwendung in ihm läuft.

2.1.1 Vorteile und Nachteile von Docker

Vorteile	**Nachteile**
Ressourcennutzung	Sicherheit
Portabilität	Verwaltung

(Quelle: Skript K.11)

2.2 Python

Python ist eine universelle, höhere Programmiersprache. Es ist eine Programmiersprache mit klarer Syntax und einfachen Strukturen. Sie lässt sich leicht mit Modulen erweitern, weswegen Python sich großer Popularität erfreut.[Sch19]
Python besitzt eine ausgesprochen große Standardbibliothek mit vielen verbreiteten Programmieraufgaben, z.B. das Lesen und Schreiben von Dateien. [Pyt] Das hier beschriebene Projekt wird mit der Python-Bibliothek tkinter umgesetzt, auf welche im Folgenden kurz eingegangen wird.

2.2.1 tkinter

tkinter ist ein Interface zwischen Python und der Tk-Grafikbibliothek. Die wichtigsten Bildschirmelemente sind in tkinter als Klassen vordefiniert. Deren Eigenschaften werden durch, meist benamte Attribute festgelegt. Es sind jeweils einige Standardmethoden definiert. [Tkia]

3 Projekt Terminkalender

In dem Projekt soll ein Terminkalender implementiert und ausgeführt werden in einem Container der Containervirtualisierungsplattform Docker. Es soll gezeigt werden, wie einfach die Erstellung eines Docker Containers und die Übertragbarkeit auf verschiedenen Betriebssysteme ist.
Dazu wird als zunächst der Terminkalender implementiert und getestet. Danach wird mit dem Dockerfile ein Image gebaut. Das erstellte Image wird auf Docker Hub bereitgestellt. Schlussendlich wird eine Bash geschrieben, die das Docker Image herunterlädt und ausführt.
Der Terminkalender soll die Funktionen haben neue Termine einzutragen, Termine zu löschen und die vorhandenen Termine einzusehen. Python wurde gewählt, da mit tkinter und in der tkinter vorhandenen Unter-Bibliotheken „calendar“ schon die Basis für die Terminkalender App existiert. tkinter ist eine sehr einfach programmierbare graphische

Oberfläche mit einer guten Dokumentation [Tkib] und vielen online verfügbaren Beispielen [Cal]. Außerdem gibt es in Docker bereits ein Python Image, das lediglich ergänzt werden musste.
Neben der Dokumentation wurde zusätzlich noch Stack Overflow bei konkreten Fragen benutzt.

4 Implementation

4.1 Terminkalender mit Python

Der Terminkalender besteht aus einem Hauptfenster mit einer Kalenderdarstellung und drei Knöpfen für den Aufruf der drei Unterfenster: neue Termin anlegen, Termin löschen und Termin ansehen.
Als erstes werden in dem Quellcode die Bibliotheken tkinter, tkinter.tkk, tkcalendar, csv, os.path aufgerufen. tkinter, tkinter.tkk und tkcalender sind für die Oberfläche zuständig. So ist tkinter.tkk eine Unter-Bibliothek von tkinter, die es ermöglicht csv-Dateien darzustellen und in ihnen zu scrollen.
Mit der tkcalendar Bibliothek wird ein Kalenderblatt dargestellt [Cal]. Die csv Bibliothek ist zum Bearbeiten von csv-Dateien nötig. In einer csv-Dateien werden tabellarisch Daten gespeichert. Mit os.path kann die Datenverwaltung des Betriebssystems benutzt werden.

```
1 from tkinter import *
2 import tkinter.ttk as ttk
3 from tkcalendar import Calendar
4 import csv
5 import os.path
```

Als nächstes wird eine Exception angelegt, welche geworfen wird, wenn in die Eingabe für die Stunden oder Minuten nicht innerhalb der vorgeben Werte (0-23 und 0-59) ist. Insgesamt wurde in dem Programm darauf geachtet die unterschiedlichsten Fehlermeldungen abzufangen. Dafür wurden nach dem Initialisieren der Grundoberfläche unterschiedlich

Fenster für Fehlermeldungen angelegt. Es gibt Fenster für die falsche Eingabe der Uhrzeit, wenn es noch keinen Kalender gibt und wenn ein nicht existierender Termin gelöscht werden soll.

```
#Exception
class ValueWrong(Exception):
    pass
```

Danach werden die Methoden zum Anlegen und Löschen eines Termines implementiert. Es gibt viele Ähnlichkeiten in den Methoden, weil beiden die Eingabewerte einlesen und überprüfen. Zusätzlich bringen beide die Uhrzeit ins richtige Format (00:00) Danach wird die csv-Datei gelesen und verändert. Dazu wird erst überprüft, ob Datei existiert. Dann wird die Veränderung durchgeführt. Dabei ist es wichtig, bei dem Aufruf der csv-Datei auf die richtigen Benutzerrechte zu achten. Mit den Schreibrechten 'w' kann die csv-Datei ausschließlich komplett neu geschrieben werden aber nicht verändern. Dazu benutzt man die Veränderungsrechten 'a', mit welchen an die Tabelle angehängt werden kann.

```
    if os.path.exists('./dockerfolder/myCal.csv'):
        with open('./dockerfolder/myCal.csv','a', newline= '') as
    mycal:
            writer = csv.writer(mycal)
            writer.writerow(appointment)
    else:
        with open('./dockerfolder/myCal.csv','w', newline= '') as
    mycal:
            writer = csv.writer(mycal)
            writer.writerow(header)
            writer.writerow(appointment)
```

Eine weitere Schwierigkeit csv-Dateien ist, dass man Elemente nicht direkt löschen kann. Deswegen wurde zum Entfernen eines Termins erst die csv-Datei in eine Liste kopiert und dann aus der Liste dieser Eintrag gesucht und entfernt, um anschließend die veränderte Liste wieder in die csv-Datei zu schreiben.

```
    try:
        with open('./dockerfolder/myCal.csv','r', newline= '') as
    mycal:
```

```
            reader = csv.reader(mycal)
            appointments = list(reader)
        with open('./dockerfolder/myCal.csv', 'w', newline= '') as
    mycal:
            writer = csv.writer(mycal)
            for y in range(len(appointments)):
                if appointments[y] == appointment:
                    appointments.pop(y)
                    writer.writerows(appointments)
                    found = True
                    break
     except FileNotFoundError:
        x.destroy()
        return error_window(basis)
```

Die csv-Datei, welche die Termineinträge enthält, wird in einem Verzeichnis abgelegt. Dieses Verzeichnis wird in dem Dockerfile als Volumen gemountet, damit die Einträge auch nach dem Schließen des Containers noch gespeichert sind.
Dann werden die drei Fenster für die unterschiedlichen Funktionen (Termineingabe, Termin löschen und Termin ansehen) angelegt. Dabei musste besonders darauf geachtet werden, dass die Fenster auf dem Grundfenster basieren und die Eingabefelder alle als Texteingabe gewählt sind.
Zum Schluss werden in dem Grundfenster drei Knöpfe angelegt, die jeweils ein Unterfenster öffnen. Mit der mainloop-Funktion des Grundfensters wird dafür gesorgt, dass das Programm jeder Zeit Eingaben entgegen nimmt.

4.2 Docker-Container

In dem unteren Quellcode ist das Dockerfile zum Erstellen des Images mit der Kalender App dargestellt. Zum Erstellen muss man im Terminal in dem Verzeichnis sein, wo das Dockerfile und die Anwendung gespeichert sind, und führt dann den Befehl *docker build -t laurahawk/calender-0108 .* aus. Der Name muss aus dem Nutzernamen bei Docker und dem Namen des Repositoriums bestehen[Doc]. Mit dem Befehl *docker push laurahawk/calender-0108* wird das Image hochgeladen.

```
FROM python:3.7
```

```
2 COPY . /app
3 RUN pip install tkcalendar
4 CMD python /app/myCAL.py
```

Die erste Zeile des Dockerfiles muss mit einer FROM Anweisung beginnen. Diese gibt das übergeordnete Image an, von dem ausgehend das Projekt aufgebaut wird. Alle nachfolgenden Befehle basieren auf diesem übergeordneten Image, das in diesem Fall das Image python ist, um das mit Python programmierte Projekt interpretieren und ausführen zu können. Die folgende COPY-Anweisung kopiert das aktuelle Verzeichnis, in welchem sich das Dockerfile befindet, und fügt es dem Dateisystem des Containers unter dem zuvor definierten Arbeitsverzeichnis hinzu. Mit der RUN-Anweisung werden noch zusätzliche Schichten dem Image hinzugefügt, in diesem Fall die Bibliothek tkcalendar. Abschließend definiert die CMD-Anweisung dem Container, was ausgeführt werden soll, um die Anwendung zu starten. In diesem Fall teilt sie Python mit, dass mycal.py ausgeführt werden soll. [Doc]
Das Verknüpfen des Bildschirms mit dem Container stellt sich als schwierig heraus., besonders da es Unterschiede bei den Betriebssystemen gibt. Das MacOS hat keinen xhost native installiert um darüber die Bildschirmfreigabe zu machen. Nach einigen Suchen im Internet fand sich eine Lösung bei der XQuartz installiert wird und als erstes gestartet wird. [Mac]
Es wurden zwei Bash Dateien programmiert für MacOS und Windows. Der einzige Unterschied ist dabei die Aufführung des XQuartz Programms (Zeile: 8-9).
Eine Bash hat immer den Befehl #!/*bin*/*bash* und das Kürzel sh. In der Bash wird zu erst das neueste Image heruntergeladen und dann mit dem Run-Befehl ausgeführt. Bei dem RUN-Befehl wird mit -v der Display verbunden, ebenso das Verzeichnis.

```
1 #!/bin/bash
2
3 echo Image wird heruntergeladen und anschliessen starten
4
5 docker pull laurahawk/calendar-0108
6
7
8 ip=$(ifconfig en0 | grep inet | awk '$1=="inet" {print $2}')
9 xhost + $ip
10
11 docker run -e DISPLAY=$ip:0 -v /tmp/.X11-unix:/tmp/.X11-unix -v
     dockerfolder:/dockerfolder laurahawk/calendar-0108
```

5 Zusammenfassung

Die Aufgabe des Projekts, welche das erfolgreiche Implementieren und Ausführen von Terminkalender in einem Container mithilfe der Containervirtualisierungsplattform Docker ist, wurde erfüllt. Das Projekt wurde mit Python und mit der tkinter-Bibliothek umgesetzt. Schwierigkeiten hat vor allem die Einbindung der graphischen Oberfläche gemacht. Besonders herausfordert war dabei, dass die Lösung vom Betriebsystems abhäng war. Es bedurfte einer längeren Internetrecherche und mehren Testläufe bevor der Container mit dem Bildschirm verbunden werden konnte.
Zusammenfassend lässt sich sagen, dass die Umsetzung des Docker-Projekts als Anfänger in der Containervirtualisierung mit Python auf jeden Fall eine bewältigbare Aufgabe ist. Sowohl zu Docker und als auch zu Python gibt es genug Dokumentation und Beispiele um hilfreiche und erfolgreiche Ansätze zu finden .
Die größter Herausforderung war tatsächlich das Ausführen des Containers, wegen der graphischen Benutzeroberfläche.
Die Vorteile von Docker mit seiner schnellen Portierung und dem wenigen Speicherplatz wurden deutlich aber genauso klar war auch der Nachteil bei dem Verwenden einer graphischen Oberfläche zu sehen. Insgesamt überwiegen die Vorteile besonders wenn man Erfahrung in dem Ausführen der Container gesammelt hat.

Literaturverzeichnis

[Cal] URL: https://www.geeksforgeeks.org/create-a-date-picker-calendar-tkinter/.

[Doc] URL: https://docs.docker.com/get-started/overview/.

[Mac] URL: https://fredrikaverpil.github.io/2016/07/31/docker-for-mac-and-gui-applications/.

[Pyt] URL: https://de.wikipedia.org/wiki/Python_(Programmiersprache).

[Sch19] Christoher Schäfer. *Schnellstart Python,Ein Einstieg ins Programmieren für MINT-Studierende,* Springer, 2019.

[Tkia] URL: https://bildungsserver.berlin-brandenburg.de/inf-python-tkinter.

[Tkib] URL: https://docs.python.org/3/library/tkinter.html.