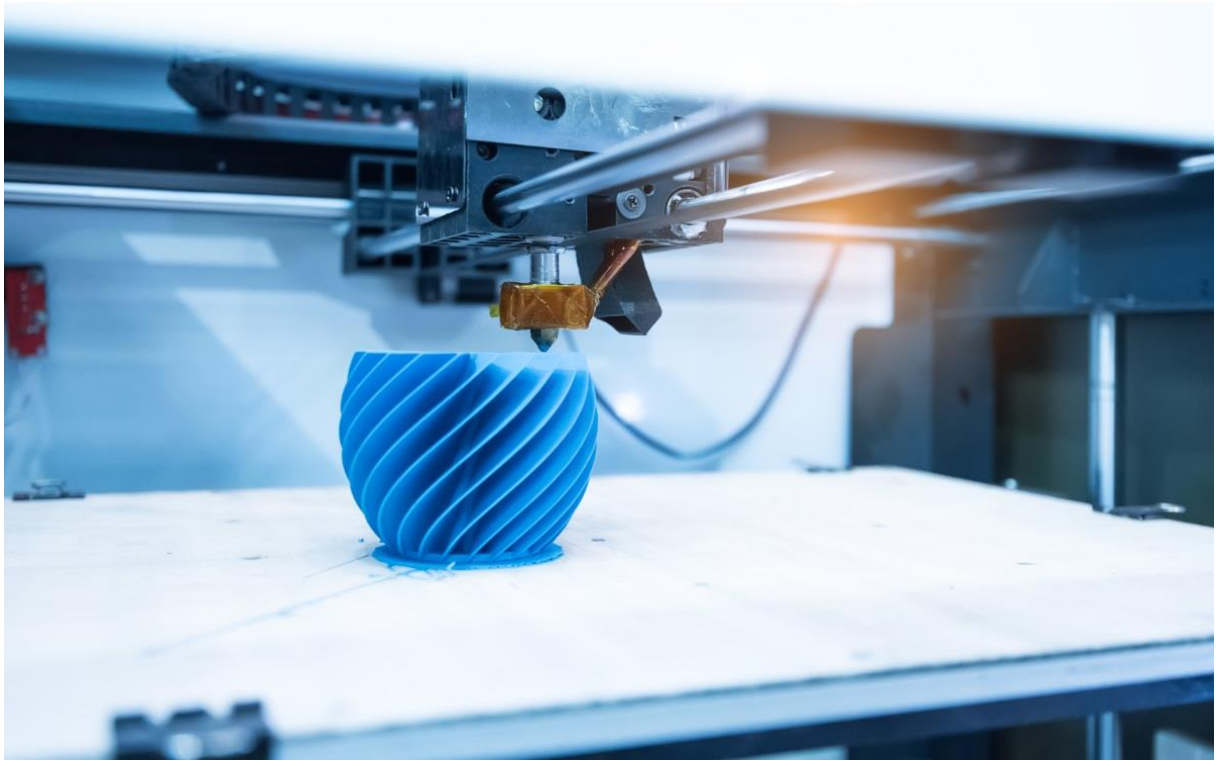
Cloud Computing – Teil 2

- Prof. Dr. Kasche –

Dokumentation

Thema: Skript zum outsourcen eines Slicing-Prozess (3D)



Kevin Lempert

Jonas G. Renz

- PIA18 -

Inhaltsverzeichnis

1 Kurzinfo

Autor:	**Kevin Lempert** **Jonas G. Renz**
Hochschuleinrichtung:	**Duale Hochschule Gera-Eisenach (Campus Gera)**
Version:	**1.0**
Programmiersprache:	**Python 3.x**
Datum:	**06.11.2020**
Systemaufruf:	**cloudslice.py**
Beschreibung:	Dieses Skript dient zum Outsourcen für Slicing-Prozesse in die Cloud. Mithilfe von AWS und Slic3r können so Slicing-Prozesse Ressourcen sparend und schnell den GCODE für 3D Drucker in AWS EC2 berechnen.

2 Bedienungsanleitung

2.1 Anleitung

Es muss für das Skript eine bestehende EC2 Instanz auf Amazon AWS existieren. Diese Instanz muss über das Betriebssystem Linux verfügen. Wenn eine keine Instanz existiert muss eine EC2 Instanz auf AWS initialisiert werden.

1. Schritt: Installieren des Programms Slic3r auf EC2-Instanz

   $ sudo apt-get install slic3r

2. Schritt: Konfiguration in der CONFIG-Sektion des Skriptes vornehmen

   2.1 Schritt: ID der Instanz eintragen

```
58  # AWS instance
59  instance_id = ['i-0dc8a2dbfad73a683']
```

   2.2 Schritt: Benutzername und den Pfad des Private-Key eintragen

```
61  # Credentials
62  cUser       = 'ubuntu'
63  cKey        = './key/Key_1.pem'
```

3. Schritt: Python-Skript ausführen

   $ py cloudslice.py

4. Schritt: Wizard mit abgefragten Parametern befüllen

5. Schritt: fertige GCODE-Datei liegt im angegebenen Verzeichnis

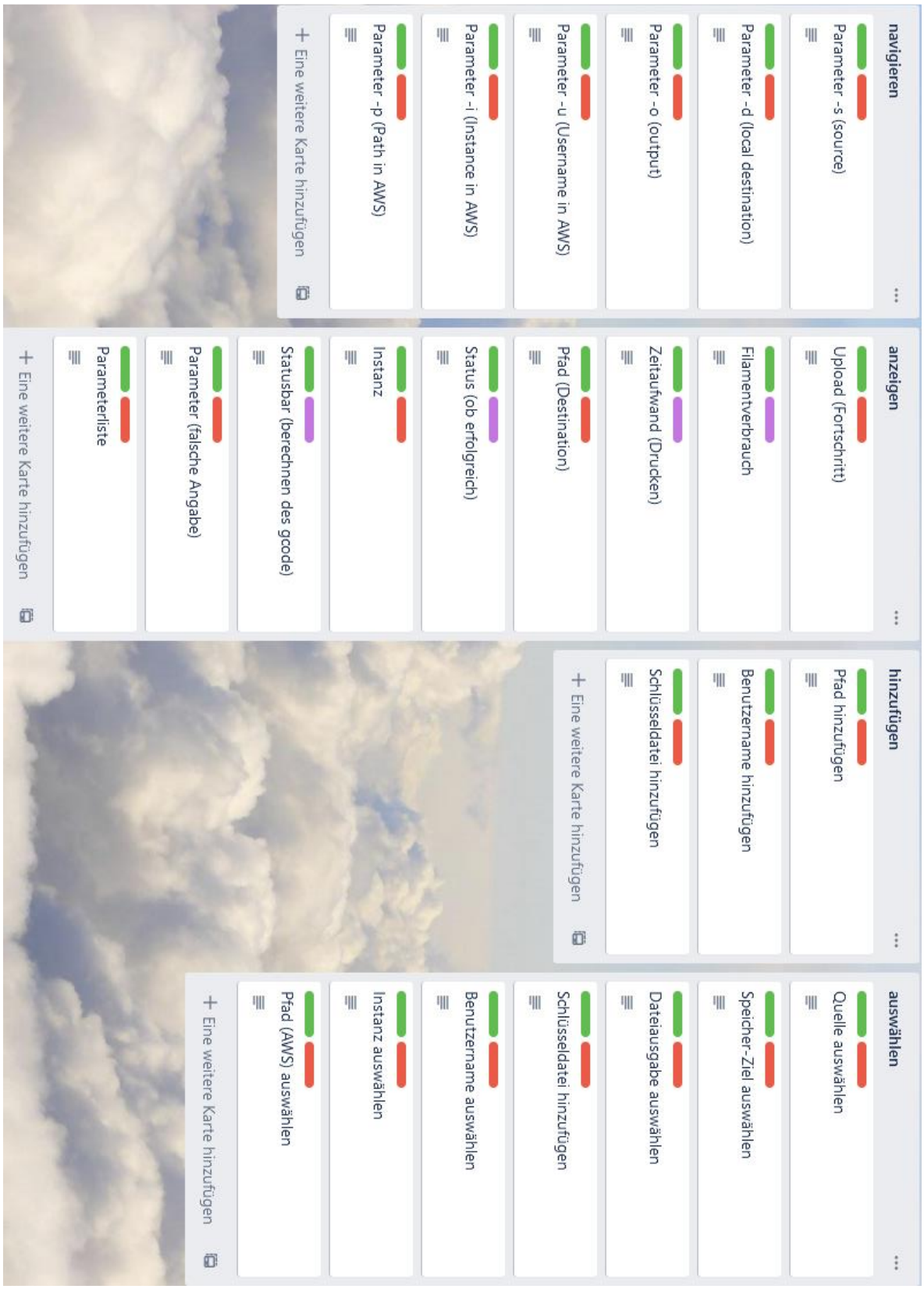
2.2 Parameter

Parameter	Variable(n)	Erklärung
--bed-temperature <t>	t = ganze Zahl, z.B.: 60	Hier soll die Druckbett-temperatur angegeben werden. Maßeinheit: [°C]
--cooling		Lüfter- und Kühlungssteuerung aktivieren
--filament-diameter <d>	d = ganze Zahl, 1.75 oder 3	Durchmesser des Filaments Maßeinheit: [mm] Default: 3
--fill-density <d>	d = ganze Zahl, z.B.: 50	Dichte der Füllung Maßeinheit: [%] Default: 40 Bereich: 0-100
--layer-height <h>	h = Kommazahl, z.B.: 0.2	Hier wird die Layer-Höhe des Modells angegeben. Maßeinheit: [mm] Default: 0.4
--nozzle-diameter <d>	d = Kommazahl, > 0.4	Durchmesser der Düse Maßeinheit: [mm] Default: 0.5
--output <o>	o = Name der Datei	Dateiname, in die gcode ausgegeben werden soll Wenn ein Verzeichnis für diese Option angegeben

		wird, wird die Ausgabe unter diesem Verzeichnis gespeichert.
--retract-length <l>	l = ganze Zahl, z.B.: 36	Länge des Rückzugs beim Anhalten der Extrusion Maßeinheit: [mm]
--scale <s>	s = Kommazahl, z.B.: 1.3 1 = Orginalgröße	Hier soll ein Skalierungsfaktor für das Model angegeben werden.
--support-material		Erzeugen von Hilfsmaterial für Überhänge
--temperature <t>	t = ganze Zahl, z.B.: 215	Temperatur Hotend 0 zum Deaktivieren Maßeinheit: [°C] Default: 200

3 Projektdokumentation

3.1 User Stories / Story-Board



Das Story Board besteht aus einzelnen Karten, in dem der Kunde seine Anforderungen äußert.

Hier ist eine Karte aus dem Bereich „anzeigen“. Die Karte enthält eine nichtfunktionale Anforderung, die von den Entwicklern eingesetzt wurde. Nicht-funktionale Anforderungen hingegen sind meist unspezifisch für ein Produkt. Beispiele für nicht-funktionale Anforderungen sind User Stories enthalten immer den gleichen Text:

Als …
möchte ich …
um …

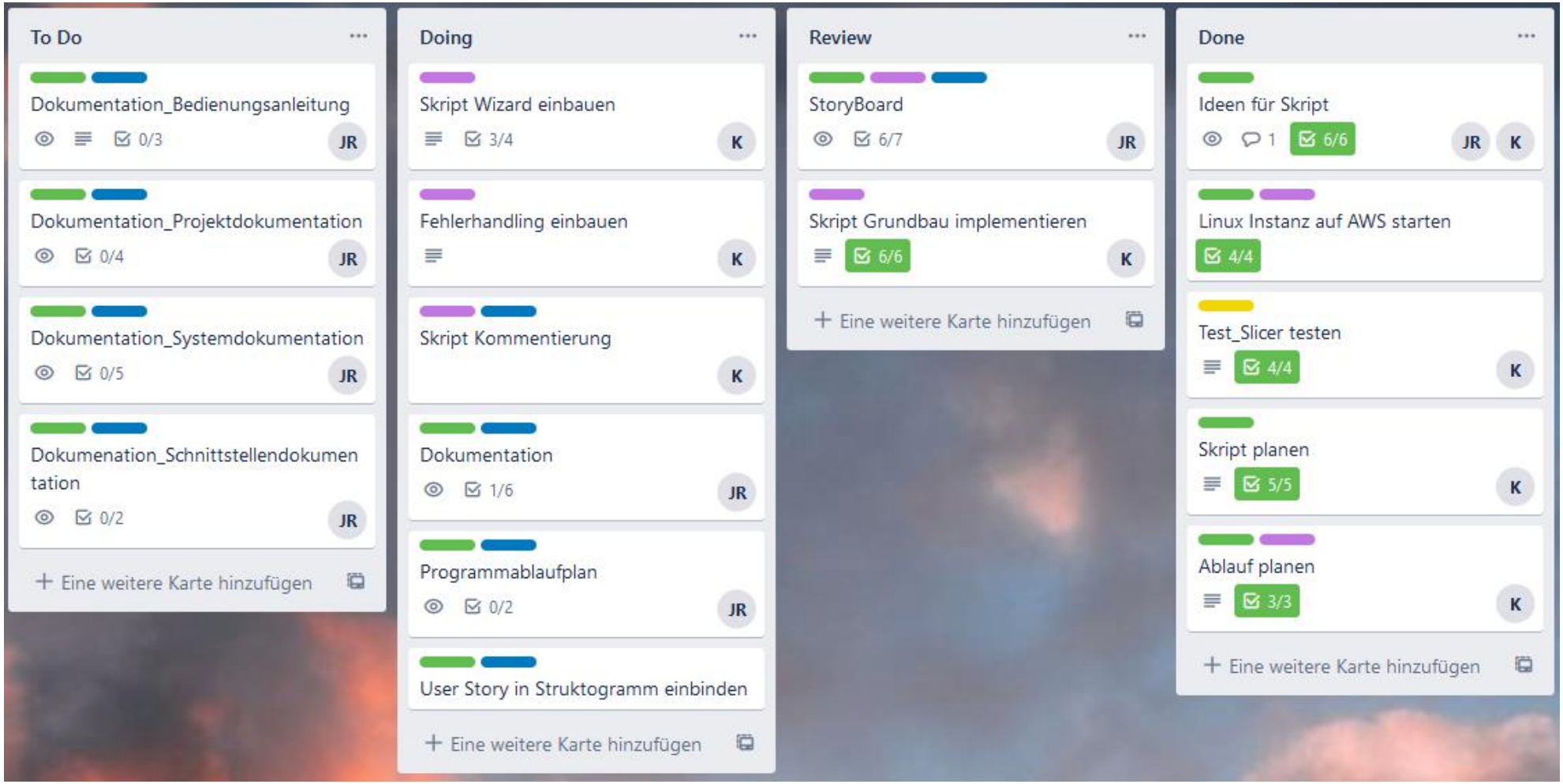
In der nächsten Karte ist eine funktionale Anforderung zu erkennen. Funktionale Anforderungen sind die Anforderungen, deren Umsetzung direkt der Zweckbestimmung des Produkts dienen. Sie sind spezifisch für dieses Produkt.

3.2 KanBan-Board

Ziel des Kanban-Boards ist es, Projektabläufe und Aufgaben zu visualisieren – in übersichtlichen Spalten mit einzelnen Einträgen, die der Reihe nach abgearbeitet werden können.

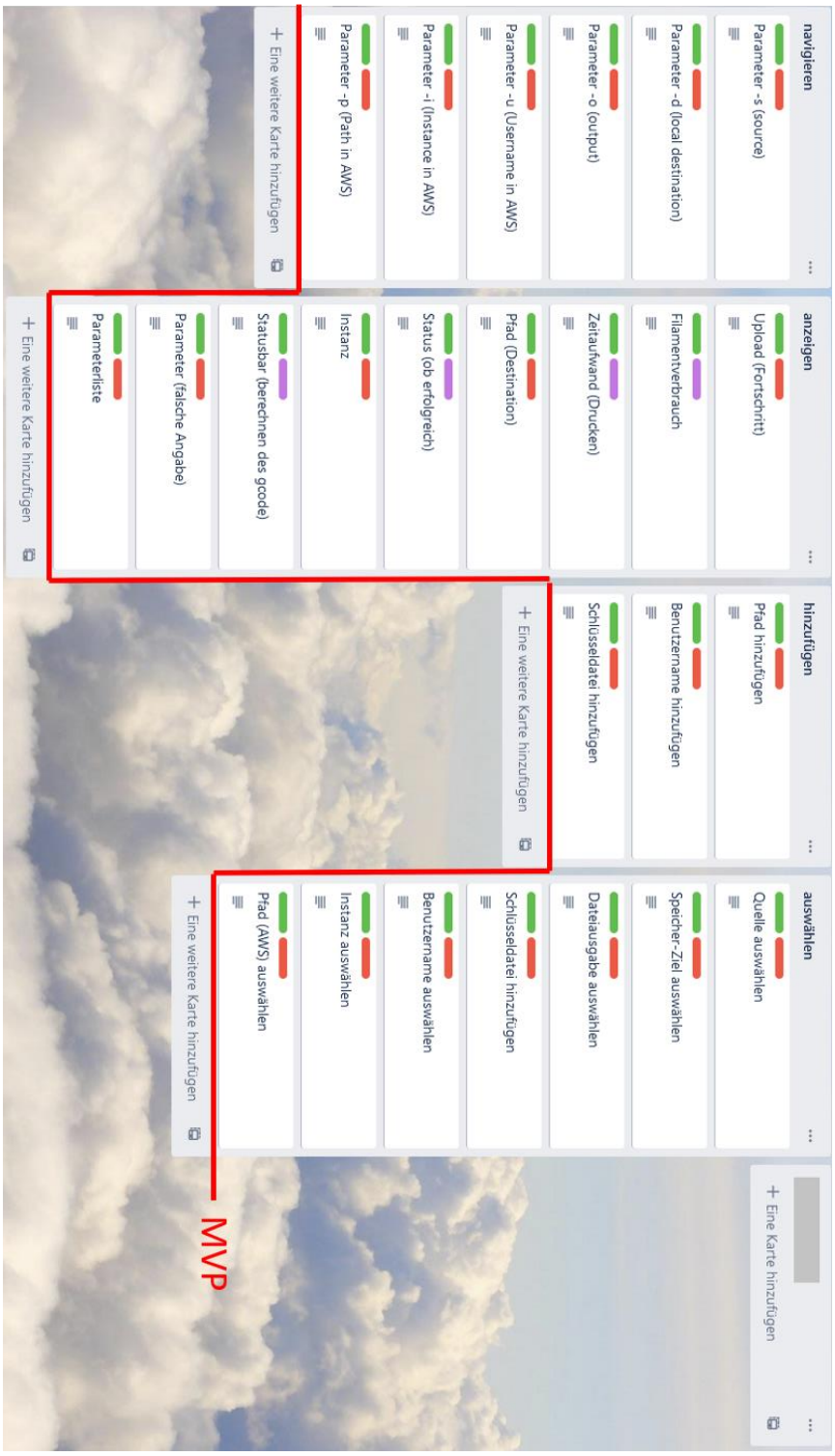
So sieht das Team auf einen Blick, was zu tun ist und in welchen Bereichen Probleme gelöst werden müssen. Dabei kann das Kanban Board auf viele verschiedene Weisen genutzt und gestaltet werden – um es optimal anzupassen und die Ergebnisse zu verbessern.

Hier ist ein kleiner Ausschnitt aus dem KanBan-Board zu sehen. Zur Übersichtlichkeit wurde aus der frühen Projektphase ein Foto übernommen.


To Do
Dokumentation_Bedienungsanleitung
0/3
JR
Dokumentation_Projektdokumentation
0/4
JR
Dokumentation_Systemdokumentation
0/5
JR
Dokumenation_Schnittstellendokumentation
0/2
JR
Eine weitere Karte hinzufügen
Doing
Skript Wizard einbauen
3/4
K
Fehlerhandling einbauen
K
Skript Kommentierung
K
Dokumentation
1/6
JR
Programmablaufplan
0/2
JR
User Story in Struktogramm einbinden
Eine weitere Karte hinzufügen
Review
StoryBoard
6/7
JR
Skript Grundbau implementieren
6/6
K
Eine weitere Karte hinzufügen
Done
Ideen für Skript
1
6/6
JR
K
Linux Instanz auf AWS starten
4/4
Test_Slicer testen
4/4
K
Skript planen
5/5
K
Ablauf planen
3/3
K
Eine weitere Karte hinzufügen

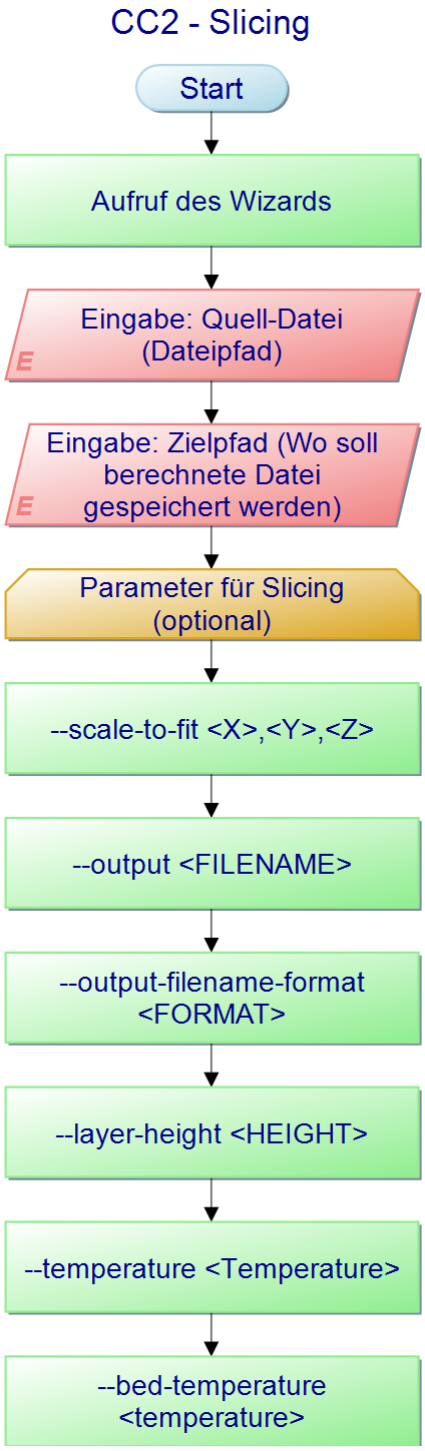
3.3 MVP

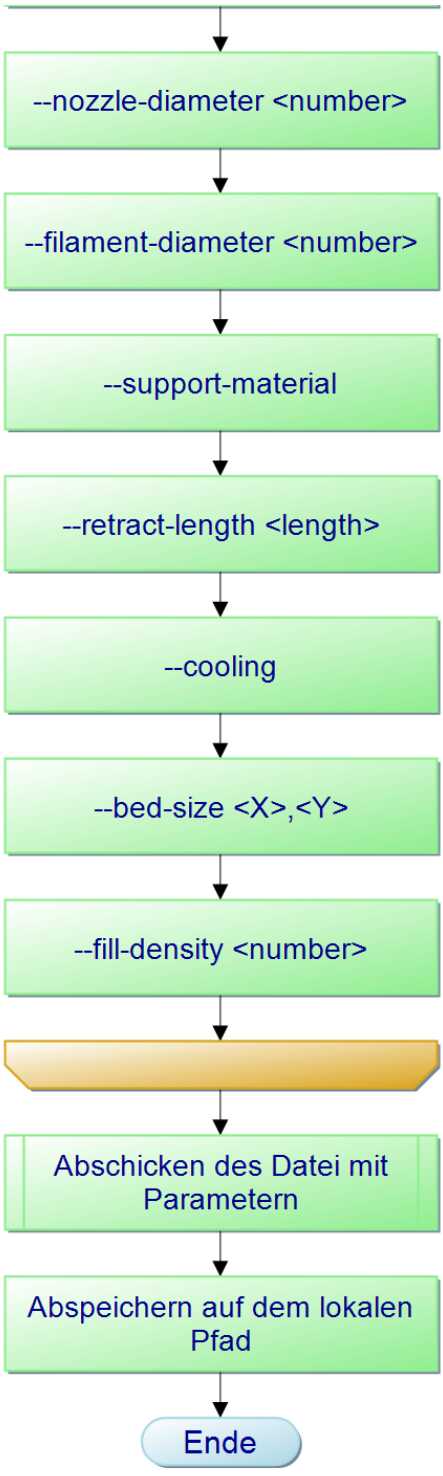
Ein Minimum Viable Product (MVP) ist die erste minimal funktionsfähige Iteration eines Produkts, das entwickelt werden muss, um mit minimalem Aufwand den Kunden-, Markt- oder Funktionsbedarf zu decken und handlungsrelevantes Feedback zu gewährleisten. Das MVP sieht ist aus dem Story-Board zu entnehmen.



4 Systemdokumentation

4.1 PAP




--nozzle-diameter <number>
--filament-diameter <number>
--support-material
--retract-length <length>
--cooling
--bed-size <X>,<Y>
--fill-density <number>
Abschicken des Datei mit Parametern
Abspeichern auf dem lokalen Pfad
Ende

5 Schnittstellendokumentation

5.1 AWS

Amazon Web Services (**AWS**) ist ein US-amerikanischer Cloud-Computing-Anbieter, der 2006 als Tochterunternehmen des Online-Versandhändlers Amazon.com gegründet wurde. Zahlreiche populäre Dienste wie beispielsweise Dropbox, Netflix, Foursquare oder Reddit greifen auf die Dienste von Amazon Web Services zurück. 2017 stufte Gartner AWS als führenden internationalen Anbieter im Cloud Computing ein.

Der Web-Service Amazon Elastic Compute Cloud (Amazon EC2) stellt sichere, skalierbare Rechenkapazitäten in der Cloud bereit. Der Service ist darauf ausgelegt, Cloud Computing für Entwickler zu erleichtern. Amazon EC2 bietet die breiteste und tiefste Computerplattform mit einer Auswahl an Prozessor, Speicher, Netzwerk, Betriebssystem und Kaufmodell.

5.2 Python SDK

Amazon stellt für Python-Skripte ein SDK zur Verfügung. Das SDK nennt sich Boto3. Boto3 erleichtert die Integration von Python-Anwendung, -Bibliothek oder Scripts in AWS-Services wie Amazon S3, Amazon EC2, Amazon DynamoDB und andere.

Boto3 hat zwei getrennte Ebenen von APIs. Die Client-APIs (oder API auf niedriger Ebene) bieten eine 1-zu-1-Zuordnung der zugrunde liegenden HTTP API-Vorgänge. Ressourcen-APIs verbergen explizite Netzwerkaufrufe und stellen stattdessen Ressourcenobjekte und Collections bereit, um auf Attribute zuzugreifen und Aktionen durchzuführen.

6 Testdokumentation

```
C:\GIT\cloudcomputing-2\script (master -> origin)
λ py cloudslice.py
  _____ _                 _  _____ _ _
 / ____| |               | |/ ____| (_)
| |    | | ___  _   _  __| | (___ | |_  ___ ___
| |    | |/ _ \| | | |/ _` |\___ \| | |/ __/ _ \
| |____| | (_) | |_| | (_| |____) | | | (_|  __/
 \_____|_|\___/ \__,_|\__,_|_____/|_|_|\___\___|
            Created by Jonas R. & Kevin L.

2020-11-06 09:31:29,329 [INFO] Starting WIZARD!
>>> File parameters
   Path input file (path/filename.stl): C:\GIT\cloudcomputing-2\script\test\cube.stl
   Path output file (path/filename.gcode): C:\GIT\cloudcomputing-2\script\test\cube.gcode

>>> Print parameters (false=default)
   > Layer height ( 0.2 - 1 )     : 0.2
   > Temperature ( 100-250 )      : 160
   > Bed temperature ( 0-100 )    : 60
   > Cooling ( true/false )       : false
   > Support ( true/false )       : false
   > Fill Density ( 0-100 )       : 20
   > Filament Diameter ( 1.75/3 ) : 1.75
   > Nozzle Diameter ( 0.4-2 )    : 0.4
   > Retract Length ( 0-10 )      : 2
   > Scale  ( 1 = orginalsize )   : 1

2020-11-06 09:32:02,208 [INFO] Start instance...
2020-11-06 09:32:22,554 [INFO] Wait for SSH server...
2020-11-06 09:32:42,721 [INFO] Server Adress: ec2-34-203-77-7.compute-1.amazonaws.com
2020-11-06 09:32:42,878 [INFO] Upload file...
2020-11-06 09:32:43,137 [INFO] Connected (version 2.0, client OpenSSH_7.6p1)
2020-11-06 09:32:43,739 [INFO] Authentication (publickey) successful!
2020-11-06 09:32:44,452 [INFO] [chan 0] Opened sftp connection (server version 3)
2020-11-06 09:32:44,903 [INFO] [chan 0] sftp session closed.
2020-11-06 09:32:44,903 [INFO] File upload complete!
2020-11-06 09:32:44,904 [INFO] Start slice process...
2020-11-06 09:32:45,127 [INFO] Connected (version 2.0, client OpenSSH_7.6p1)
2020-11-06 09:32:45,687 [INFO] Authentication (publickey) successful!
2020-11-06 09:32:47,125 [INFO] ['=> Processing triangulated mesh\n', '=> Generating perimeters\n', '=> Preparing infill\n', '=> Infilling
layers\n', '=> Generating skirt\n', '=> Exporting G-code to /home/ubuntu/temp.gcode\n', 'Done. Process took 0 minutes and 0.598 seconds\n'
, 'Filament required: 2109.2mm (5.1cm3)\n']
2020-11-06 09:32:47,126 [INFO] Slice process complete!
2020-11-06 09:32:47,126 [INFO] Download GCODE...
2020-11-06 09:32:47,353 [INFO] Connected (version 2.0, client OpenSSH_7.6p1)
2020-11-06 09:32:47,921 [INFO] Authentication (publickey) successful!
2020-11-06 09:32:48,519 [INFO] [chan 0] Opened sftp connection (server version 3)
2020-11-06 09:32:49,783 [INFO] [chan 0] sftp session closed.
2020-11-06 09:32:49,783 [INFO] GCODE download complete! File Path: C:\GIT\cloudcomputing-2\script\test\cube.gcode
2020-11-06 09:32:49,784 [INFO] Stop instance...
```

Abb. 1: Testablauf eines Slicing-Vorgangs mit CloudSlice

7 Restaufgaben

Restaufgaben stehen in den Post-Releases des KanBan-Boards. Diese Liste ist Links des Boards zu sehen.

8 Quellcode

```
"""
################################################
                SCRIPT INFORMATION
################################################
## Author: Kevin Lempert, Jonas Renz
## Copyright: Copyright 2020, CloudSlice
## Version: 1.0.0

################################################
                STARTING GUIDE
################################################

1. init EC2 instance (Linux)
2. install slic3r
    > sudo apt-get install slic3r
3. configurate the script (CONFIG section)
    > instance_id = Instance ID
    > cUser = Username
    > cKey = Private Key
    (optional)
    > u_rPath = AWS upload location (STL)
    > d_rPath = AWS download location (GCODE)

################################################
                DEV INFORMATION
################################################
AWS States:
    0 : pending
    16 : running
    32 : shutting-down
    48 : terminated
    64 : stopping
    80 : stopped

scp:
    scp -i .\key\Key_1.pem file.stl ubuntu@[instancedns]:/home/ubuntu/file.stl

ssh:
    ssh -i .\key\Key_1.pem ubuntu@[instancedns]

slice:
    slic3r file.stl --output file.gcode [options]
"""

from time import sleep
from datetime import datetime
import sys
import boto3
from botocore.exceptions import ClientError
import os
import paramiko
import logging


#--------------------------------------------------------------------------------
#                                  CONFIG
#--------------------------------------------------------------------------------

# AWS instance
instance_id = ['i-0dc8a2dbfad73a683']

# Credentials
cUser       = 'ubuntu'
cKey        = './key/Key_1.pem'

# upload/download Path AWS
u_rPath  = '/home/ubuntu/temp.stl'
d_rPath  = '/home/ubuntu/temp.gcode'

# Print parameter list for wizard
# add new parameters here
```

```
# Input text wizard : cmd slic3r
cmddict = {
  "> Layer height ( 0.2 - 1 )     " : "--layer-height",
  "> Temperature ( 100-250 )      " : "--temperature",
  "> Bed temperature ( 0-100 )    " : "--bed-temperature",
  "> Cooling ( true/false )       " : "--cooling",
  "> Support ( true/false )       " : "--support-material",
  "> Fill Density ( 0-100 )       " : "--fill-density",
  "> Filament Diameter ( 1.75/3 ) " : "--filament-diameter",
  "> Nozzle Diameter ( 0.4-2 )    " : "--nozzle-diameter",
  "> Retract Length ( 0-10 )      " : "--retract-length",
  "> Scale  ( 1 = orginalsize )   " : "--scale"
}

#--------------------------------------------------------------------------------
#                                  GLOBAL
#--------------------------------------------------------------------------------

# List for commands
cmdlist = list()

# init boto3
ec2 = boto3.resource('ec2')

# Logging Configuration
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s [%(levelname)s] %(message)s",
    handlers=[
        logging.FileHandler("debug.log"),
        logging.StreamHandler()
    ]
)

#--------------------------------------------------------------------------------
#                                  MAIN
#--------------------------------------------------------------------------------

# main function
def main():

    header()

    startwizard()

    stime_ssh = 20

    # First State Check
    if( getinstancesstate() == 80 ):
        logging.info("Start instance...")
        startinstance()
    elif(getinstancesstate() == 16):
        logging.info("Instance already running...")
        stime_ssh = 5
    else:
        logging.info("Wait for instance...")
        while True:
            sleep(2)
            if(getinstancesstate() == 16 or getinstancesstate() == 80):
                if(getinstancesstate() == 80):
                    startinstance()
                break

    # Wait for instance start
    while True:
        sleep(3)
        if(getinstancesstate() == 16):
            logging.info("Wait for SSH server...")
            sleep(stime_ssh)
            break

    # Get instance DNS
    server = getinstancesDNS()
    logging.info("Server Adress: " + server)

    if( getinstancesstate() == 16 ):
```

```
        # SCP upload:
        logging.info("Upload file...")
        putSCP(server , u_lPath, u_rPath)
        logging.info("File upload complete!")

        # SSH command:
        command = buildcommand()
        logging.info("Start slice process...")
        makeSSH(server, command)
        logging.info("Slice process complete!")

        # SCP download:
        logging.info("Download GCODE...")
        getSCP(server, d_lPath, d_rPath)
        logging.info("GCODE download complete! File Path: " + d_lPath)

        # Stop instance
        logging.info("Stop instance...")
        stopinstance()
    else:
        logging.error("Error! Instance is stopped!")

#--------------------------------------------------------------------------------
#                                  WIZARD
#--------------------------------------------------------------------------------

# wizard for print parameters
def startwizard():

    global u_lPath
    global d_lPath

    logging.info("Starting WIZARD!")
    print(">>> File parameters")

    while(True):
        u_lPath   = input("   Path input file (path/filename.stl): ")
        if( os.path.isfile(u_lPath) ):
            break
        else:
            logging.error("File " + u_lPath + " doesnt exist!")

    d_lPath = input("   Path output file (path/filename.gcode): ")

    print("\n>>> Print parameters (false=default)")
    for key, item in cmddict.items():
        value = input("   " + key + ": ")
        if(value.lower() in "true"):
            cmdlist.append( item )
        elif(value.lower() not in "false"):
            cmdlist.append( item + " " + value )

    print()

# build slic3r command
def buildcommand():
    command = 'slic3r ' + u_rPath + ' --output ' + d_rPath
    for value in cmdlist:
        command = command + " " + value
    return command

#--------------------------------------------------------------------------------
#                                  FUNCTIONS
#--------------------------------------------------------------------------------

#==================================AWS===================================

# start instance
def startinstance():
    try:
        response = ec2.instances.filter(InstanceIds=instance_id).start()
    except ClientError as e:
        logging.error(e)

# stop instance
def stopinstance():
```

```
    try:
        response = ec2.instances.filter(InstanceIds=instance_id).stop()
    except ClientError as e:
        logging.error(e)

# get instance state
def getinstancesstate():
    instances = ec2.instances.filter(InstanceIds=instance_id)
    for instance in instances:
        return(instance.state.get("Code"))

# get DNS from instance
def getinstancesDNS():
    instances = ec2.instances.filter(InstanceIds=instance_id)
    for instance in instances:
        return(instance.public_dns_name)


#=================================SCP=================================

# SCP - upload
def putSCP(server, lPath, rPath):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(server, username=cUser, key_filename=cKey )
    sftp = ssh.open_sftp()
    sftp.put(lPath, rPath)
    sftp.close()
    ssh.close()

# SCP - download
def getSCP(server, lPath, rPath):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(server, username=cUser, key_filename=cKey )
    sftp = ssh.open_sftp()
    sftp.get(rPath,lPath)
    sftp.close()
    ssh.close()

#=================================SSH=================================

# start SSH connection
def makeSSH(server, command):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(server, username=cUser, key_filename=cKey )
    stdin, stdout, stderr = ssh.exec_command(command)
    lines = stdout.readlines()
    logging.debug(str(lines))
    ssh.close()


#-----------------------------------------------------------------------------
#                             PRINT-INFOS
#-----------------------------------------------------------------------------

# Print CMD header
def header():

        print("  _____ _                  _  _____ _ _           ")
        print(" / ____| |               | |/ ____| (_)          ")
        print("| |    | | ___  _   _  __| | (___ | |_  ___ ___  ")
        print("| |    | |/ _ \| | | |/ _` |\___ \| | |/ __/ _ \ ")
        print("| |____| | (_) | |_| | (_| |____) | | | (_|  __/ ")
        print(" \_____|_|\___/ \__,_|\__,_|_____/|_|_|\___\___| ")
        print("             Created by Jonas R. & Kevin L.      ")
        print("")

#-----------------------------------------------------------------------------
#                             START
#-----------------------------------------------------------------------------

# start point
if __name__ == '__main__':
   main()
```