

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Курсова робота**

з дисципліни: «Компоненти програмної інженерії»

на тему: «Програмний додаток системи файлів, що міститься у Docker-контейнері, розроблення інтерфейсу командного рядка та тести написані з використанням Robot фреймворку»

Виконав

студент IІІ курсу

групи КП-01

Тітов Єгор Павлович

Перевірив

Погорєлов Володимир Володимирович

Київ 2023

**Код**

| **api.py** |
| --- |
| from flask import Flask, request  from flask\_restful import Resource, Api  from Directory import Directory  from BinaryFile import BinaryFile  from LogTextFile import LogTextFile  from BufferFile import BufferFile  import json  app = Flask(\_\_name\_\_)  api = Api(app)  parentDirectory = Directory('parentDir')  fileName = 'binary file'  content = 'binary content'  binary = BinaryFile(fileName, content, parentDirectory)  name = 'buffer file'  size = 10  buffer = BufferFile(name, size)  maxElements = 10  name = 'directory'  directory = Directory(name, maxElements)  name = 'logTextFile'  log = LogTextFile(name, parentDirectory)  class BinaryApi(Resource):  def \_\_init\_\_(self):  self.binary = binary  def get(self):  return self.binary.read()  def post(self):  data = request.get\_json()  parentDir = Directory(data["parent"])  self.binary = BinaryFile(data["fileName"], data["content"], parentDir)  return {'message': 'BinaryFile is successfully created'}  def put(self):  data = request.get\_json()  parentDir = Directory(data["parent"])  return self.binary.move(parentDir)  def delete(self):  return self.binary.delete()  class BufferApi(Resource):  def \_\_init\_\_(self):  self.buffer = buffer  def get(self):  return self.buffer.consume()  def post(self):  data = request.get\_json()  parentDir = Directory(data["parent"])  self.buffer = BufferFile(data["fileName"], data["maxSize"], parentDir)  return {'message': 'BufferFile is successfully created'}  def put(self):  data = request.get\_json()  parentDir = Directory(data["parent"])  return self.buffer.move(parentDir)  def patch(self):  data = json.loads(request.data)  return self.buffer.push(data["element"])  def delete(self):  return self.buffer.delete()  class DirectoryApi(Resource):  def get(self):  return {'hello': 'world'}  def \_\_init\_\_(self):  self.directory = directory  def post(self):  data = request.get\_json()  self.directory = Directory(data["name"], data["maxElements"])  return {'message': 'Directory is successfully created'}  def put(self):  data = request.get\_json()  parentDir = Directory(data["parent"])  return self.directory.move(parentDir)  def delete(self):  return self.directory.delete()  class LogTextApi(Resource):  def \_\_init\_\_(self):  self.logText = LogTextFile  def get(self):  return self.logText.read()  def post(self):  data = request.get\_json()  parentDir = Directory(data["parent"])  self.logText = LogTextFile(data["fileName"], parentDir)  return {'message': 'LogTextFile is successfully created'}  def put(self):  data = request.get\_json()  parentDir = Directory(data["parent"])  return self.logText.move(parentDir)  def patch(self):  data = request.get\_json()  return self.logText.log(data["line"])  def delete(self):  return self.logText.delete()  api.add\_resource(BinaryApi, '/binaryfile')  api.add\_resource(BufferApi, '/bufferfile')  api.add\_resource(DirectoryApi, '/directory')  api.add\_resource(LogTextApi, '/logtextfile')  if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True) |
| **cli.py** |
| from BinaryFile import BinaryFile  from BufferFile import BufferFile  from LogTextFile import LogTextFile  from Directory import Directory  import sys  import pip.\_vendor.requests as requests  n = len(sys.argv)  Request = "http://localhost:8888/" + sys.argv[2] + "?"  for i in range(3, n):  Request += sys.argv[i]  if i != n-1:  Request += "&"  response = ""  if sys.argv[1] == "get":  response = requests.get(Request)  if sys.argv[1] == "post":  response = requests.post(Request)  if sys.argv[1] == "patch":  response = requests.patch(Request)  if sys.argv[1] == "delete":  response = requests.delete(Request)  print("Response:", response.json()) |
| **Robot.robot** |
| Library OperatingSystem  \*\*\* Variables \*\*\*  ${cli} /Projects/qa/qa\_kp01\_Titov/lab2/cli.py  \*\*\* Test Cases \*\*\*  Dir create  ${result} = Run Process python3 ${cli} post directory name\=root max\_elems\=100  Should Contain ${result.stdout} Status code: 200  Dir move  ${result} = Run Process python3 ${cli} patch directory name\=name parent\=root  Should Contain ${result.stdout} Status code: 200  Dir delete  ${result} = Run Process python3 ${cli} delete directory name\=name  Should Contain ${result.stdout} Status code: 200  Dir read  ${result} = Run Process python3 ${cli} get directory name\=name parent\=root  Should Contain ${result.stdout} Status code: 200 |

**Висновки**

Метою курсової роботи було розроблення системи файлів, розташованої у Docker-контейнері, розроблення інтерфейсу командного рядка та його тестування за допомогою Robot framework.