|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ | | | | | | | | | |
|  |  | | | |  | |  | |
| Пермский государственный национальный  исследовательский университет | | | | | | | | | |
|  | |  | | | |  | |
| ОТЧЕТ  по лабораторной работе №1  «Создание клиентского приложения по выполнению CRUD-операций на популярном информационном ресурсе»  по дисциплине  «Технологии разработки распределенных приложений» | | | | | | | |
|  | | |  |  | | | |
| Работу выполнил студент гр. КМБ-16  Дровосеков Даниил Алексеевич  12.11.2020 | | |  | Проверил  доцент кафедры прикладной математики и информатики,  к. ф.-м.н., доц.  Деменев Алексей Геннадьевич | | | |
|  | | |  |  | | | |
| Пермь, 2020 г. | | | | | | | | | |

**Введение**

**Цель**: Изучение возможностей использования API современных популярных информационных ресурсов внешними приложениями.

**Задача**: реализация приложения, удовлетворяющего требованиям.

**Требования**:

1. Разработанное приложение должно предоставлять возможности создания, чтения, обновления и удаления некоторых объектов выбранного информационного ресурса (выполнения CRUD операций).

2. CRUD операции должны выполняться при помощи API выбранного ресурса, причём по крайней мере один тип операции должен требовать аутентификации на нём.

3. Должна быть возможность выполнения по крайней мере двух CRUD операций для одного и того же объекта выбранного информационного ресурса.

**Постановка задачи**

Требуется спроектировать, реализовать, протестировать и оценить приложение согласно таблице №1:

***Распределение баллов за выполнение работы (***Максимальное количество баллов — 10)***:***

Таблица 1. Требования к выполнению работы

| **Требование к заданию** | **Максимальное количество баллов** |
| --- | --- |
| Приложение позволяет выполнять CRUD‑операции над объектами выбранного ресурса | 1 (за каждый тип операции) |
| Приложение позволяет выполнять по крайней мере две CRUD‑операции для одного и того же объекта выбранного ресурса | 1 |
| Приложение выполняет по крайней мере одну из CRUD‑операций после успешной аутентификации | 2 |
| Приложение не требует повторной аутентификации при перезапуске программы | 1 |
| Параметры успешной аутентификации сохраняются при перезапуске программы в зашифрованном виде | 2 |

**Проектирование**

В качестве информационного ресурса был выбран веб-сервис для хостинга IT-проектов “GitHub” [1-3]. В качестве объектов информационного ресурса выступают текстовые файлы в репозитории. Функция создания (CREATE) объекта информационного ресурса реализуется в виде создание текстового файла в репозитории, название файла и его содержание вводится пользователем. Функция чтения (READ) объекта информационного ресурса реализуется в виде просмотри содержимого текстового файла и его названия из репозитория, либо просмотре названий всех файлов в репозитории. Функция удаления (DELETE) объекта информационного ресурса реализуется в виде удаления текстового файла из репозитория. Функция модификации (UPDATE) объекта информационного ресурса реализуется в виде изменения содержимого текстового файла в репозитории.

Для реализации приложения был выбран язык программирования Python 3.7 [4,5].

Применялась среда разработки Jupyter Notebook 7.2 [6,7]. Приложение работает с REST API v3 GitHub [2,3] с помощью python-библиотеки “github” [8]. Шифрование содержимого файла, который хранит логин и токен пользователя, осуществляется с помощью библиотеки “pycrypto” [9], с использованием симметричного шифрования AES с ключем 16 байт. Для получения строки байт используется модуль “binascii” [11]. Для реализации ввода пароля, с защитой от визуального съема информации, используется модуль “getpass” [10]. Для форматирования данных полученных с помощью API, используется модуль “base64” [12].

**Сценарии использования программы**

Сценарий использования №1 - авторизация

1. Авторизация пользователя, который ранее не авторизовался
2. Получения списка файлов
3. Выход из приложений
4. Повторный вход того же пользователя
5. Получение списка файлов

Ожидаемый результат: повторный вход пользователя без авторизации, успешное получение списка файлов и сохранение данных авторизации в зашифрованном виде

Сценарий использования №2 – создание нового файла в репозитории

1. Вход пользователя, авторизованного ранее
2. Получение списка файлов
3. Создание нового файла
4. Получение списка файлов

Ожидаемый результат: появление имени созданного файла в списке файлов

Сценарий использования №3 – просмотр файла из репозитория

1. Вход пользователя, авторизованного ранее
2. Получение списка файлов
3. Просмотр содержимого текстового файла из репозитория

Ожидаемый результат: просмотр содержимого выбранного текстового файла

Сценарий использования №4 – изменение содержимого файла в репозитории

1. Вход пользователя, авторизованного ранее
2. Получение списка файлов
3. Изменение содержимого файла в репозитории
4. Просмотр содержимого текстового файла из репозитория

Ожидаемый результат: изменение содержимого выбранного файла

Сценарий использования №5 - удаление

1. Вход пользователя, авторизованного ранее
2. Получение списка файлов
3. Удаление файла из репозитория
4. Получение списка файлов

Ожидаемый результат: исчезновение имени удалённого файла из списка файлов

**Тестирование**

Сценарий №1

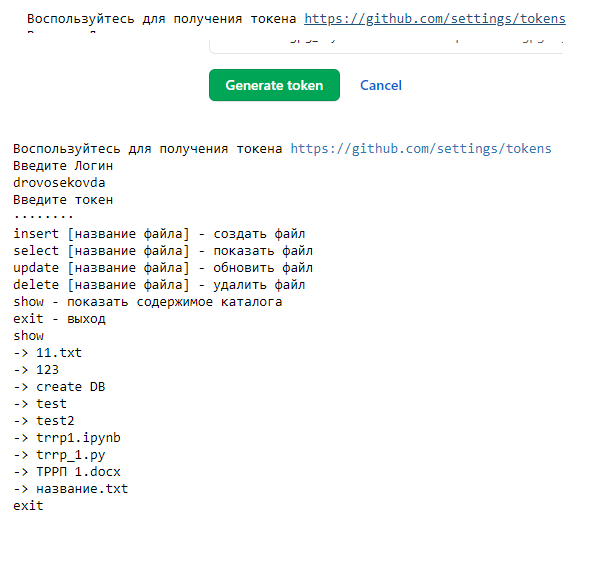


Рис. 1 Авторизация пользователя, просмотр файлов.

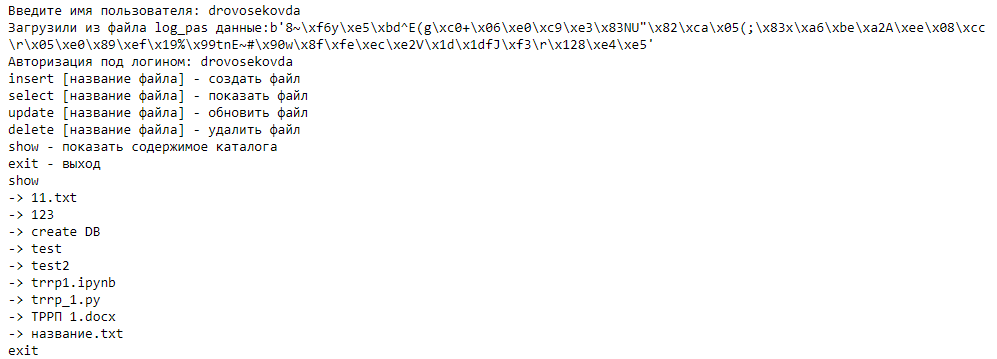


Рис. 2 Просмотр файлов пользователем авторизовавшимся ранее.

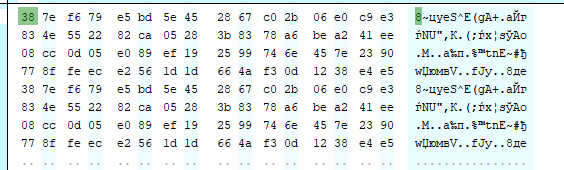


Рис. 3 Содержимое файла, хранящего логин и пароль

Результат: Успешный повторный вход пользователя без авторизации, сохранение данных в зашифрованном виде и просмотр файлов.

Сценарий №2

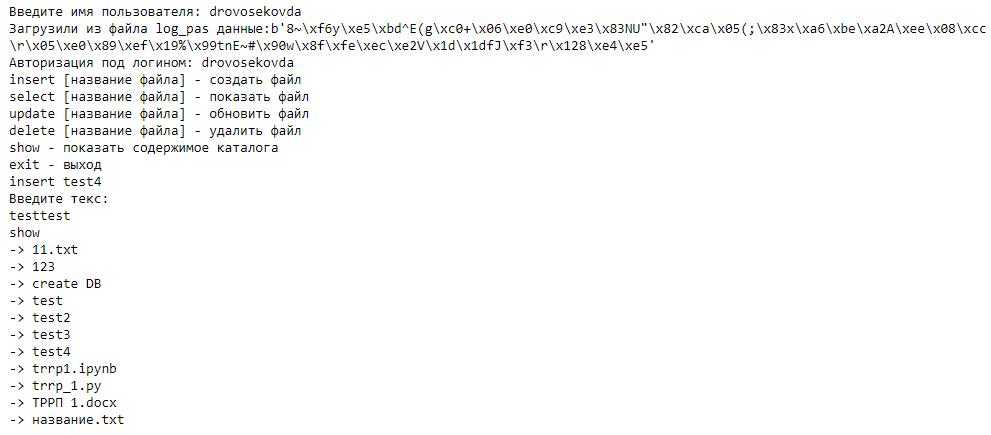


Рис. 4 Создание файла.

Результат: Файл был успешно создан

Сценарий №3

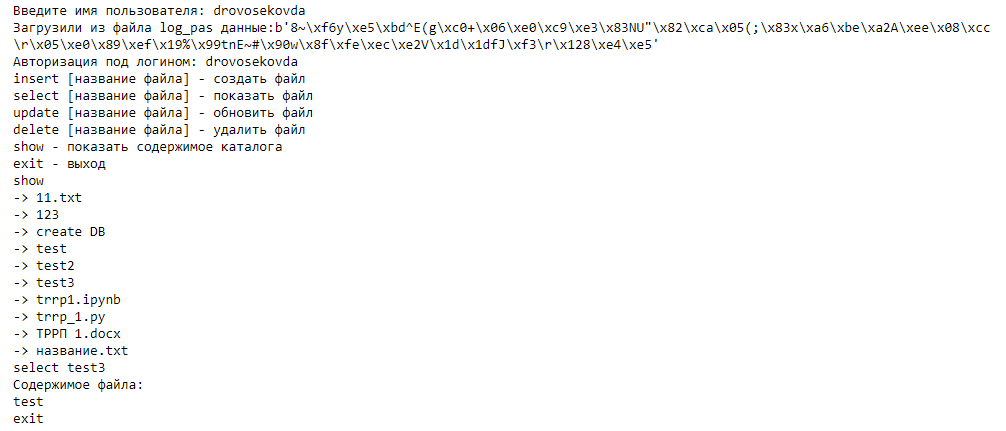


Рис. 5 Просмотр содержимого файла

Результат: Содержимое файла было успешно получено.

Сценарий №4

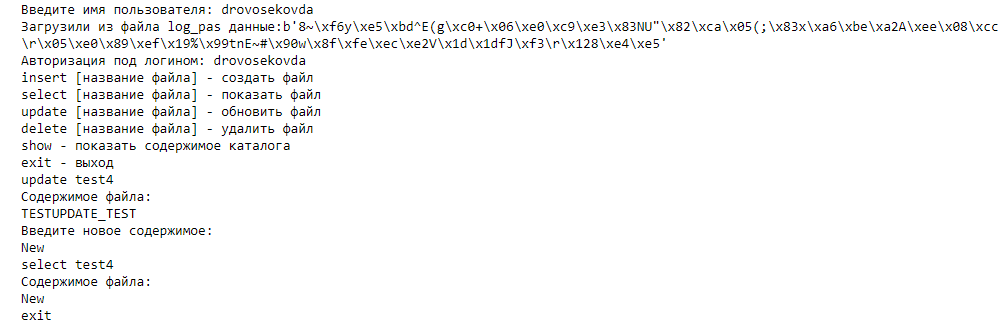


Рис. 6 Изменение содержимого файла.

Результат: Удалось успешно изменить содержимое текстового файла.

Сценарий №5

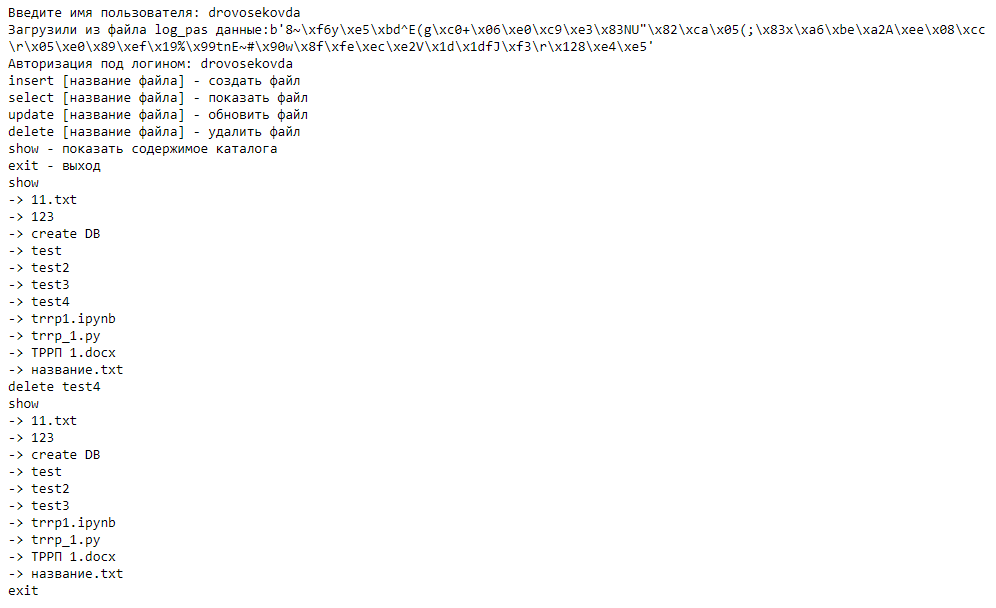


Рис. 7 Удаление файла.

Результат: Удалось успешно удалить файл.

**Заключение**

В результате работы была достигнута цель и выполнены следующие требования:

* Требование «Приложение позволяет выполнять CRUD операции над объектами выбранного ресурса» выполнено полностью. Ожидаемое количество баллов: 4.
* Требование «Приложение позволяет выполнять по крайней мере две CRUD операции для одного и того же объекта выбранного ресурса» выполнено полностью. Ожидаемое количество баллов: 1.
* Требование «Приложение выполняет по крайней мере одну из CRUD операций после успешной аутентификации» выполнено полностью. Ожидаемое количество баллов: 2.
* Требование «Приложение не требует повторной аутентификации при перезапуске программы» выполнено полностью. Ожидаемое количество баллов: 1.
* Требование «Параметры успешной аутентификации сохраняются при перезапуске программы в зашифрованном виде» выполнено полностью. Ожидаемое количество баллов: 2.

**Перечень использованных источников**

1. GitHub. url: <https://github.com/> (дата обращения: 12.11.2020)
2. GitHub Terms of Service. url: <https://docs.github.com/en/free-pro-team@latest/github/site-policy/github-terms-of-service> (дата обращения: 12.11.2020)
3. Site policy. url: <https://docs.github.com/en/free-pro-team@latest/github/site-policy> (дата обращения: 12.11.2020)
4. Python 3.7.0. url: <https://www.python.org/downloads/release/python-370/> (дата обращения: 12.11.2020)
5. History and License. url: <https://docs.python.org/3.7/license.html> (дата обращения: 12.11.2020)
6. Installation. url: <https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html> (дата обращения: 12.11.2020)
7. Documentation. url: <https://jupyter.org/documentation> (дата обращения: 12.11.2020)
8. Introduction. url: <https://pygithub.readthedocs.io/en/latest/introduction.html> (дата обращения: 12.11.2020)
9. Project description. url: <https://pypi.org/project/pycrypto/> (дата обращения: 12.11.2020)
10. Getpass — Portable password input. url: <https://docs.python.org/3/library/getpass.html> (дата обращения: 12.11.2020)
11. Binascii — Convert between binary and ASCII. url: <https://docs.python.org/3/library/binascii.html> (дата обращения: 12.11.2020)
12. base64 — Base16, Base32, Base64, Base85 Data Encodings. url: <https://docs.python.org/3/library/base64.html> (дата обращения: 12.11.2020)

**Приложение**

Код программы:

|  |
| --- |
| import getpass  import binascii  from Crypto.Cipher import AES  from github import Github  import base64  key = binascii.unhexlify('aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa')  cipher = AES.new(key)  def encrypt\_(username,password):  if len(username)<16:  for i in range(0,16-len(username)):  username+=" "  pas1=password[0:16]  pas2=password[16:32]  pas3=password[32:40]  if len(pas3)<16:  for i in range(0,16-len(pas3)):  pas3+=" "  tag1 = cipher.encrypt(username)  tag21 = cipher.encrypt(pas1)  tag22 = cipher.encrypt(pas2)  tag23 = cipher.encrypt(pas3)  file\_name="log\_pas"  with open(file\_name, 'ab') as byte\_file:  byte\_file.write(tag1)  byte\_file.write(tag21+tag22+tag23)  username=""  password=""  b=0  file\_name="log\_pas"  try:  with open(file\_name, 'rb') as byte\_file:  username1=input("Введите имя пользователя: ")  tag=byte\_file.readline()  while (tag != None and password==""):  print("Загрузили из файла "+file\_name+" данные:" +str(tag))  tag1 = cipher.decrypt(tag)  username2=tag1.decode()  username=username2.split()[0]  if (username==username1):  password=username2.split()[1]  print("Авторизация под логином: "+username)  else:  tag=byte\_file.readline()  except:  print("")  if (str(password)==""):  print("Воспользуйтесь для получения токена"+" "+"https://github.com/settings/tokens")  print("Введите Логин")  username=input()  print("Введите токен")  password=getpass.getpass()  b=1    #print(username)  try:  g = Github(password)  if b==1:  encrypt\_(username,password)  repo = g.get\_repo("drovosekovda/test")#Получить репозиторий  print("insert [название файла] - создать файл")  print("select [название файла] - показать файл")  print("update [название файла] - обновить файл")  print("delete [название файла] - удалить файл")  print("show - показать содержимое каталога")  print("exit - выход")  command=input()  while command!="exit":  try:  if (command.split()[0]=="insert"):  print("Введите текст:")  text=input()  repo.create\_file(command.split()[1], "",text)#Создать файл  if (command=="show"):  #Получить файлы в репозитории  repo = g.get\_repo("drovosekovda/test")#Получить репозиторий  contents = repo.get\_contents("")  for content\_file in contents:  print("->",content\_file.path)  if (command.split()[0]=="update"):  contents = repo.get\_contents(command.split()[1])#Получить файл по названию  file\_bytes = base64.b64decode(contents.content)  file\_str = file\_bytes.decode('utf-8')  print("Содержимое файла:")  print(file\_str)  print("Введите новое содержимое:")  text=input()  repo.update\_file(contents.path, "", text, contents.sha)#Изменить  if (command.split()[0]=="delete"):  contents = repo.get\_contents(command.split()[1])#Получить файл по названию  repo.delete\_file(contents.path, "", contents.sha)#Удалить файл  if (command.split()[0]=="select"):  contents = repo.get\_contents(command.split()[1])#Получить файл по названию  file\_bytes = base64.b64decode(contents.content)  file\_str = file\_bytes.decode('utf-8')  print("Содержимое файла:")  print(file\_str)  except:  print("Error")  command=input()  except:  print("Ошибка авторизации") |