МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАСШИРЕНИЕ УТИЛИТЫ РЕЗЕРВНОГО КОПИРОВАНИЯ ДАННЫХ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Шаймардановой Анар Галиевны

Научный руководитель
зав. кафедрой, к. ф.-м. н., доцент _____ С. В. Миронов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Disk Uploader — утилита резервного копирования данных [1]. Disk Uploader является уникальным проектом, поскольку не было найдено готовых решений, позволяющих сохранять backup's баз данных CRM (Customer Relationship Management) или других учетных систем по расписанию в популярные облачные хранилища.

Данная утилита может помочь различным учетным системам регулярно сохранять backup-ы баз данных автоматически. Также можно настроить некоторые параметры для утилиты, такие как минимальное и максимальное возможное количество файлов в каталоге. При нехватке места на диске может производиться удаление старых файлов до минимально возможного количества. При каждом запуске приложения может производиться нотификация пользователя на почту об успешной или неудачной загрузке файла.

С помощью утилиты можно выполнять резервное копирование любых файлов. В Disk Uploader уже реализована возможность работы с такими облачными хранилищами, как Яндекс.Диск и Dropbox. Но в настоящее время в нашей стране имеется тенденция к использованию облачных хранилищ Облако Mail.ru и Yandex Object Storage, поэтому было решено расширить утилиту Disk Uploader и добавить возможность работы с данными хранилищами.

Целью настоящей работы является расширение существующей утилиты Disk Uploader, добавление возможности работы с такими облачными хранилищами, как Облако Mail.ru и Yandex Object Storage. Для достижения поставленной цели необходимо выполнить следующие задачи:

- изучить библиотеку Sardine языка Java для работы с Облаком Mail.ru по протоколу WebDAV;
- изучить способ формирования подписи для аутентификации с Yandex Object Storage;
- изучить библиотеки языка Java, необходимые для работы с Yandex Object Storage;
- добавить функционал для возможности вызова утилиты из другого приложения на языке Java;
- продемонстрировать работу утилиты с помощью создания задачи в «Планировщике заданий».

Выпускная квалификационная работа состоит из трех глав. Работа вклю-

чает в себя 82 страницы, 38 библиографических источников. Первая глава называется «Утилита Disk Uploader» и включает в себя описание утилиты Disk Uploader. Вторая глава носит название «Технологии взаимодействия с облачными хранилищами» и включает описание технологий, используемых при работе с хранилищами. Третья глава называется «Расширение Disk Uploader для работы с сервисами Облако Mail.ru и Yandex Object Storage» и включает описание кода, реализованного в работе, а также включает демонстрацию реализованного функционала.

1 Утилита Disk Uploader

Перечислим основные достоинства Disk Uploader:

- работа с такими облачными хранилищами, как Яндекс.Диск и Dropbox;
- вызов утилиты через командную строку;
- возможность использования таких параметров загрузки файла, как минимальное и максимальное количество файлов в каталоге в хранилище;
- удаление файлов при превышении указанного максимально допустимого количества до минимально допустимого;
- удаление файлов при нехватке места в хранилище до указанного минимально допустимого количества;
- добавление возможности нотификации пользователя о статусе загрузки файла на email при настройке .xml файла;
- логирование работы утилиты;
- относительная «легкость» утилиты, так как для работы с хранилищами по протоколам HTTP и WebDAV использовались библиотеки Jersey и Sardine, т. е. никаких библиотек, разработанных специально для работы с конкретным хранилищем, например, AWS SDK for Java, не используется;

В первую очередь утилита разрабатывалась для сохранения backup-ов баз данных учетных систем по расписанию. Кроме того, Disk Uploader подходит для загрузки любых файлов в облачные хранилища. Вызвать утилиту по расписанию можно, используя сервис crontab в Linux или приложение «Планировщик заданий» в Windows.

Таким образом, утилита Disk Uploader является оптимальным решением для сохранения backup-ов баз данных CRM систем.

2 Технологии взаимодействия с облачными хранилищами

2.1 Технологии Облака Mail.ru

Облако Mail.ru— хранилище данных в интернете. Файлы в облаке доступны в любой точке мира, на любых устройствах. Загрузка и скачивание файлов выполняется за считанные секунды, все файлы в облаке находятся под надежной защитой [2].

К облаку Mail.ru возможно подключение по протоколу WebDAV.

2.1.1 Протокол WebDAV

Протокол WebDAV дает возможность веб-серверу вести себя также, как и файловый сервер, поддерживая совместную разработку веб-контента [3].

Протокол WebDAV расширяет стандартный набор HTTP-методов и заголовков для того, чтобы предоставить возможность создание файла или папки, редактирование, копирование, перемещение, удаление файла и т. д. [3].

2.1.2 Библиотека Sardine

Sardine— библиотека нового поколения для работы по протоколу WebDAV на Java. Цель данной библиотеки состоит в том, чтобы предоставить наибольшее число методов для работы с WebDAV сервером [4].

`DavResource`— класс, включенный в библиотеку Sardine. Объекты данного класса используются для описания ресурса на удаленном сервере. Объектом может быть каталог или файл [5].

2.2 Технологии Yandex Object Storage

Рассмотрим технологии, использующиеся при работе с Yandex Object Storage.

Yandex Object Storage— универсальное масштабируемое решение для хранения данных. Данный сервис подходит для любых проектов: как требующих надежного и быстрого доступа к данным, так и проектам, не имеющим таких требований [6].

Для того, чтобы передавать файлы в Yandex Object Storage, необходимо правильно составить запрос к сервису. Для практически любого запроса требуется заголовок Authorization. Заголовок Authorization используется для аутентификации запроса и включает в себя подпись [7]. Рассмотрим шаги, необходимые для формирования подписи.

Создание Canonical Request

Первый шаг — создание canonical request. Далее будем его обозначать как канонический запрос. Формат такого запроса в общем виде выглядит следующим образом:

```
<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>
```

где

`HTTPMethod` — один из HTTP методов(например, GET, PUT, HEAD, DELETE и т. д.);

`CanonicalURI` — закодированный в формате URL абсолютный путь к компоненту URI;

`CanonicalQueryString` определяет закодированные в формате URL параметры запроса;

`CanonicalHeaders` — это список заголовков запросов и их значений. Пары имя-значения разделяются символом переноса строки;

`SignedHeaders` — это отсортированный по алфавиту, разделенный точкой с запятой список приведенных к нижнему регистру имен заголовков запроса;

`HashedPayload` — это шестнадцатеричное значения хэша SHA256 полезной нагрузки запроса.

Создание String To Sign

Следующий шаг: вычисление String to Sign. Далее будем обозначать ее как строка для подписи.

Строка для подписи включает в себя метаинформацию про запрос и про канонический запрос, вычисленный на предыдущем шаге [8].

Чтобы создать строку для подписи, необходимо сконкатенировать используемый алгоритм, дату и время, `CredentialScope` и `SHA-256` хэш канонического запроса. В общем виде строка для подписи выглядит следующим образом:

```
StringToSign =
    Algorithm + \n +
    RequestDateTime + \n +
    CredentialScope + \n +
    HashedCanonicalRequest
```

Рассмотрим метод создания строки для подписи:

1. первая строка строки для подписи— используемый алгоритм для формирования хэша, который мы используем в каноническом запросе. Если мы используем алгоритм SHA-256, то используемый алгоритм— AWS4-HMAC-SHA256;
2. следующая строка строки для подписи— дата запроса в формате ISO8601. Это тот же формат, который используется в заголовке x-amz-date вида YYYYMMDD'T'HHMMSS'Z', где
   - YYYY— год;
   - MM— месяц;
   - DD— день;
   - HH— час;
   - MM— минута;
   - SS— секунда.
3. следующая строка— значение credential scope. Данное значение— строка, включающая дату, название региона, в которому мы обращаемся, название сервиса, к которому мы обращаемся, и строка *"aws4_request"*;
4. завершающая строка — хэш канонического запроса. Хэшированный канонический запрос должен быть приведен к нижнему регистру и закодирован в шестнадцатеричном формате [8].

Создание Signature

Следущий шаг — формирование подписи. Для формирования подписи необходимо выполнить следующее:

1. формируем signing key. Для этого мы используем секретный ключ, используемый для доступа к хранилищу, и создаем несколько HMACs (hash-based message authentication codes или кодов аутентификации) [9]. Чтобы получить signing key, необходимо выполнить следующие действия:

   ```
   kSecret = your secret access key
   ```

```
kDate = HMAC("AWS4" + kSecret, Date)
kRegion = HMAC(kDate, Region)
kService = HMAC(kRegion, Service)
kSigning = HMAC(kService, "aws4_request")
```

где

а) `HMAC(key, data)` — применение функции `HMAC-SHA256` к `data` (данным, сообщению) с ключом `key`;

б) `kSecret` — секретный ключ;

в) `kDate` получается в результате применения функции `HMAC`, которой в качестве ключа передается конкатенация строк `AWS4` и `kSecret`, а в качестве данных— дата. Дату необходимо передавать в формате YYYYMMDD;

г) `kRegion` получается в результате вызова `HMAC` с параметрами `kDate` и названием региона;

д) `kService` присваивается вызов `HMAC` с параметрами `kRegion` и названием сервиса;

е) `kSigning` определяется вызовом `HMAC` с параметрами `kService` и строкой `"aws4_request"`.

2. Чтобы получить подпись, необходимо использовать полученный signing key и строку для подписи, полученную на предыдущем шаге. Мы должны вызвать функцию `HMAC-SHA256` с перечисленными параметрами, и затем перевести результат в шестнадцатеричный формат:

```
signature = HexEncode(HMAC(derived signing key, string to sign))
```

Добавление подписи к HTTP запросу

После вычисления подписи, ее необходимо добавить к запросу. Рассмотрим способ добавления подписи к запросу с помощью HTTP заголовка Authorization.

Следующий псевдокод показывает содержание заголовка Authorization [10]:

```
Authorization: algorithm Credential=access key ID/credential scope,
↪ SignedHeaders=SignedHeaders, Signature=signature
```

2.2.1 Используемые HTTP заголовки

Далее рассмотрим заголовки HTTP запросов, которые использовались в практической части.

- Authorization— информация, необходимая для авторизации;
- Content-Length— длина тела запроса (не включая заголовки);
- Expect — это ожидаемый код. Устанавливается в 100-continue.
- Host— хост-получатель запроса;
- x-amz-date— дата и время на устройстве, осуществляющим запрос. Дата задается в формате Thu, 18 Jan 2018 09:57:35 GMT [11];
- x-amz-content-sha256— при использовании подписи версии 4, данный заголовок включает в себя хэш полезной нагрузки запроса [12].

2.2.2 Используемые средства языка Java

Рассмотрим основные Java библиотеки, используемые при работе с Yandex Object Storage.

Для создания HTTP запросов используется библиотека Jersey.

Была использована библиотека commons-codec. Программное обеспечение Apache Commons Codec предоставляет реализации наиболее распространенных способов шифрования и дешифрования данных, таких как Base64, Hex, Phonetic и URL [13].

Класс `DigestUtils` упрощает операции класса `MessageDigest` [14].

В этом классе нас интересует метод `public static String sha256Hex(String data)`. Вызывая данный метод, мы передаем строку `data`. Он вычисляет `SHA-256` хэш для нее и возвращает шестнадцатеричное представление хэша [14].

Пакет javax.crypto предоставляет классы и интерфейсы для криптографических операций [15]. Далее рассмотрим класс `Mac` данного пакета.

Класс `Mac` предоставляет функциональные возможности MAC (Message Authentication Code) алгоритма или код аутентификации сообщения или имитовставка [16].

Следующий метод:

```
public final void init(Key key, AlgorithmParameterSpec params) throws
↪ InvalidKeyException, InvalidAlgorithmParameterException
```

выполняет инициализацию объекта `Mac`, вызывающего данный метод с переданным ключом `key` и параметрами `params` [16].

3 Расширение Disk Uploader для работы с сервисами Облако Mail.ru и Yandex Object Storage

Разработка кода выполнялась в среде IntelliJ IDEA [17], на языке Java версии 1.8. Работа выполнялась на компьютере с процессором Intel Core i5, с оперативной памятью 8 Гб.

Опишем план, который необходимо выполнить для достижения поставленной цели дипломной работы:

1. реализовать классы, необходимые для взаимодействия с сервисами Облако Mail.ru и Yandex Object Storage;
2. расширить существующие классы, добавив несколько переменных и методов, благодаря которым утилита сможет взаимодействовать с Cloud Mail.ru и Yandex Object Storage;
3. реализовать функционал, который позволит вызвать утилиту из другого приложения на языке Java.

3.0.1 Взаимодействие с сервисами Облако Mail.ru и Yandex Object Storage

Рассмотрим реализацию необходимого функционала для добавления возможности работы утилиты с сервисами Облако Mail.ru и Yandex Object Storage.

Для работы с данными хранилищами прежде всего необходимо реализовать следующие методы:

1. `public void authorize(String username, String password)` — получить
   данные для авторизации;
2. `public DiskInfo getInfoOfDisk() throws Exception` — получить информацию о диске;
3. `public long getFreeSpaceSize() throws Exception` — получить
   информацию о свободном пространстве на диске;
4. `public long getTotalSpaceSize() throws Exception` — получить информацию об общем количестве памяти на диске;
5. `public boolean isFolderExist(String folderName)`
   `throws Exception` —
   проверить, существует ли данный каталог;
6. `public DiskResultOperation uploadFile(String folderName,`
   `String fileName, File file) throws Exception` — загрузить файл в

каталог;

7. `public DiskResultOperation deleteFile(String folderName, String fileName) throws Exception` — удалить файл из каталога;
8. `public DiskResultOperation deleteListOfFiles(List<FileInfo> listOfFilesToDelete, String folderName) throws Exception` — удалить список файлов;
9. `public int getNumOfFiles(String folderName) throws Exception` — получить количество файлов в каталоге;
10. `public List<FileInfo> getListOfFiles(String folderName) throws Exception` — получить список файлов каталога.

Для взаимодействия с Облаком Mail.ru и с Yandex Object Storage были реализованы все вышеперечисленные методы в классах `DiskActionMailRu` и `DiskActionYandexObjectStorage` соответственно.

Расширены существующие классы утилиты, необходимые для того, чтобы утилита смогла взаимодействовать с классами `DiskActionMailRu` и `DiskActionYandexObjectStorage`. Были расширены следующие классы: `TypeDiskEnum`, `CmdParser`, `CmdResultParser`, `DiskFactory`, `DiskManager`.

Был реализован класс `DiskParameters`. Объект данного класса можно создать в другом проекте на Java, задать параметры вызова утилиты и вызвать соответствующий метод `public static boolean upload(DiskParameters diskParameters)` класса `DiskManager`, передав в качестве параметра созданный объект класса `DiskParameters`.

3.1 Инструкция пользователю

Чтобы запустить утилиту через командную строку, необходимо сформировать исполняемый файл .jar и запустить его с необходимыми параметрами. Перечислим возможные параметры:

1. `--td` — тип диска. YA — Яндекс Диск, DB — Dropbox, MRC — Cloud Mail.ru, YOS — Yandex Object Storage;
2. `--fu` — путь к загружаемому файлу;
3. `--tr` — максимальное количество файлов, которое можно загрузить в каталог;
4. `--mr` — минимальное количество файлов, которое можно загрузить в каталог;
5. `--cu` — каталог в хранилище, куда мы хотим загрузить файл;

6. `--at` — аутентификационный токен, необходимый для доступа к хранилищу;
7. `--un` — имя пользователя, необходимое для доступа к Cloud Mail.ru;
8. `--pw` — пароль, необходимый для доступа к Cloud Mail.ru;
9. `--kid` — key id или идентификатор ключа, необходимый для доступа к Yandex Object Storage;
10. `--sac` — secret access key или секретный ключ, необходимый для доступа к Yandex Object Storage.

Например, загрузка файла File.txt в Yandex Object Storage будет выглядеть следующим образом:

```
java -jar DiskUploader-1.0-jar-with-dependencies.jar --td=YOS
↪  --fu=C:\Users\user\Downloads\File3.txt --cu=test-bucket3 --kid=...
↪  --sac=... --tr=3 --mr=1
```

Также можно создать задачу в приложении «Планировщик заданий» с необходимыми параметрами для регулярного вызова утилиты.

ЗАКЛЮЧЕНИЕ

Все задачи, поставленные в дипломной работе, а именно:

- изучить библиотеку Sardine языка Java для работы с Облаком Mail.ru по протоколу WebDAV;
- изучить способ формирования подписи для аутентификации с Yandex Object Storage;
- изучить библиотеки языка Java, необходимые для работы с Yandex Object Storage;
- добавить функционал для возможности вызова утилиты Disk Uploader из другого приложения на языке Java;
- продемонстрировать работу утилиты Disk Uploader с помощью создания задачи в «Планировщике заданий».

были выполнены. Цель была достигнута.

Утилита Disk Uploader автоматизирует процесс сохранения файлов, в том числе backup-ов баз данных учетных систем в облачные хранилища.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Disk Uploader — File Uploading Tool for Dropbox and Yandex.Disk [Электронный ресурс]. — URL: https://github.com/pchelicam/DiskUploader (Дата обращения 19.05.2021). Загл. с экр. Яз. англ.

2 Что такое Облако? [Электронный ресурс]. — URL: https://help.mail.ru/cloud_web/about/what (Дата обращения 03.03.2021). Загл. с экр. Яз. рус.

3 WebDAV [Электронный ресурс]. — URL: https://www.oslogic.ru/knowledge/27/webdav/ (Дата обращения 03.03.2021). Загл. с экр. Яз. рус.

4 UsageGuide [Электронный ресурс]. — URL: https://github.com/lookfirst/sardine/wiki/UsageGuide (Дата обращения 03.03.2021). Загл. с экр. Яз. англ.

5 sardine/DavResource.java [Электронный ресурс]. — URL: https://github.com/lookfirst/sardine/blob/master/src/main/java/com/github/sardine/DavResource.java (Дата обращения 09.03.2021). Загл. с экр. Яз. англ.

6 Yandex Object Storage [Электронный ресурс]. — URL: https://cloud.yandex.ru/docs/storage/ (Дата обращения 11.03.2021). Загл. с экр. Яз. рус.

7 Signature Calculations for the Authorization Header: Transferring Payload in a Single Chunk (AWS Signature Version 4) [Электронный ресурс]. — URL: https://docs.aws.amazon.com/AmazonS3/latest/API/sig-v4-header-based-auth.html (Дата обращения 11.03.2021). Загл. с экр. Яз. англ.

8 Task 2: Create a string to sign for Signature Version 4 [Электронный ресурс]. — URL: https://docs.aws.amazon.com/general/latest/gr/sigv4-create-string-to-sign.html (Дата обращения 11.03.2021). Загл. с экр. Яз. англ.

9 Task 3: Calculate the signature for AWS Signature Version 4 [Электронный ресурс]. — URL: https://docs.aws.amazon.com/general/latest/gr/sigv4-calculate-signature.html (Дата обращения 15.03.2021). Загл. с экр. Яз. англ.

10 Task 4: Add the signature to the HTTP request [Электронный ресурс]. — URL: https://docs.aws.amazon.com/general/latest/gr/sigv4-add-signature-to-request.html (Дата обращения 15.03.2021). Загл. с экр. Яз. англ.

11 Общие заголовки запросов [Электронный ресурс]. — URL: https://cloud.yandex.ru/docs/storage/s3/api-ref/common-request-headers (Дата обращения 18.03.2021). Загл. с экр. Яз. рус.

12 Common Request Headers [Электронный ресурс]. — URL: https://docs.aws.amazon.com/AmazonS3/latest/API/RESTCommonRequestHeaders.html (Дата обращения 18.03.2021). Загл. с экр. Яз. англ.

13 Apache Commons Codec [Электронный ресурс]. — URL: https://commons.apache.org/proper/commons-codec/ (Дата обращения 18.03.2021). Загл. с экр. Яз. англ.

14 Class DigestUtils [Электронный ресурс]. — URL: https://commons.apache.org/proper/commons-codec/apidocs/org/apache/commons/codec/digest/DigestUtils.html (Дата обращения 18.03.2021). Загл. с экр. Яз. англ.

15 Package javax.crypto [Электронный ресурс]. — URL: https://docs.oracle.com/javase/7/docs/api/javax/crypto/package-summary.html (Дата обращения 24.03.2021). Загл. с экр. Яз. англ.

16 Class Mac [Электронный ресурс]. — URL: https://docs.oracle.com/javase/7/docs/api/javax/crypto/Mac.html (Дата обращения 24.03.2021). Загл. с экр. Яз. англ.

17 IntelliJ IDEA [Электронный ресурс]. — URL: https://www.jetbrains.com/ru-ru/idea (Дата обращения 20.05.2021). Загл. с экр. Яз. англ.