|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
|  | | | |
|  | |  |  |
| МИНОБРНАУКИ РОССИИ | | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"Московский государственный университет информационных технологий, радиотехники и**  **электроники"**  **МИРЭА** | | | |
| Институт информационных технологий | | | |
| Кафедра информатики и информационных систем | | | |

|  |  |
| --- | --- |
| **КУРСОВАЯ РАБОТА** | |
| **по дисциплине** | |
| **«Объектно-ориентированное программирование»** | |
| 1. **Тема курсовой работы «**Компьютерная модель синхронизации аудио на базе открытого интерфейса социальной сети «ВКонтакте»»   **Вариант № 131225** | |
| Студент группы ИСБО-01-13 | *Адушкин А.А.* |
| Руководитель курсовой работы | *Хлебников А.А.* |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_2015 г. |  |
|  |  |  |
| «Допущен к защите» | «\_\_»\_\_\_\_\_\_\_2015 г. |  |

Москва 2015

|  |  |
| --- | --- |
|  | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждениевысшего профессионального образования"Московский государственный технический университет информационных технологий, радиотехники и электроники"МИРЭА | |
| Факультет информационных технологий (ИТ) | |
| Кафедра информатики и информационных систем (ИИС) | |
|  | **Утверждаю** |
|  | Заведующий  кафедрой Соловьев Игорь Владимирович |
|  | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_2015 г. |
| **ЗАДАНИЕ** | |
| **на выполнение курсовой работы** | |
| **по** **дисциплине** «Объектно-ориентированное программирование» | |

**Студент** Адушкин Антон Андреевич **Группа** ИСБО-01-13

**1. Тема** «Компьютерная модель синхронизации аудио на базе открытого интерфейса социальной сети «ВКонтакте»».

1. **Исходные данные:** документация API «ВКонтакте», Java JDK 1.8, библиотеки gson и Retrofit, целевая файловая система.
   1. **Перечень вопросов, подлежащих разработке, и обязательного графического материала:**
   * Разработать модуль расширения на базе Java Extensible Applications
   * Разработать окружение тестирования
   * Разработать абстрактные классы для обращения к API
   * Разработать абстрактные классы для хранения полученной информации
   * Изучение работы протоколов HTTP

|  |  |  |  |
| --- | --- | --- | --- |
| 1. **Срок представления к защите курсовой работы:** **до** «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_\_ г. | | | |
|  | | | |
| Задание на курсовой  проект (работу) выдал | «\_\_\_»\_\_\_\_\_\_\_\_\_ 201\_\_г. |  | *Хлебников А. А.* |
| Задание на курсовой  проект (работу) получил | «\_\_\_»\_\_\_\_\_\_\_\_\_\_ 201\_\_ г. |  | *Адушкин А.А.* |

**ОТЗЫВ**

на курсовой проект студента группы ИСБО-01-13 *Адушкина А.А.*

по дисциплине «*Объектно-ориентированное программирование*» на тему

*«***Компьютерная модель синхронизации аудио на базе открытого интерфейса социальной сети «ВКонтакте»»**

(вариант № 131225)

УДК 004

ББК 32.97

А 328

**Рецензент: Хлебников А.А.**

**Адушкин А.А.** **Курсовая работа по дисциплине «Объектно-ориентированное программирование» на тему: «Создание компьютерной модели синхронизации аудио на базе открытого интерфейса социальной сети «ВКонтакте»»**

**/**руководитель Хлебников А.А. / М.: МИРЭА, 2015. -

Курсовая работа включает в себя описание разрабатываемой программы, задачей которой является синхронизация аудио, описание теоретической части и перечень вопросов к разработке.

Course work includes a description of a program under development, whose task is to synchronize audio, the description of a theoretical part and a list of questions to develop.

**Список основных сокращений**

**IRC** (Internet Relay Chat) – протокол прикладного уровня для обмена сообщениями в режиме реального времени.

**ICQ** (I Seek You) – централизованная служба мгновенного обмена сообщениями сети Интернет.

**OSI** (Open Systems Interconnection), сетевая модель – базовая эталонная модель взаимодействия открытых систем.

**TCP/IP** – набор сетевых протоколов передачи данных.

**ПК** – персональный компьютер.

**ID** – идентификационный номер пользователя.

**Реферат**

В курсовой работе выполнено поэтапное проектирование программы по предложенному заданию, разработан интерфейс для общения с пользователем.

Курсовая работа имеет стандартный вид и включает в себя введение, основную часть, заключение, список источников и приложение.

Во Введении сформулирована цель и задачи работы на основе анализа исходных данных.

В основной части пояснительной записки приведены теоретические сведения о функциональных средствах, используемых в разработке, а также перечень вопросов к разработке и пример работы программы.

В заключительной части пояснительной записки дана оценка результатам разработки.

**Article summary**

In the course work step-by-step design of the program on the proposed assignment has done, the interface has designed to communicate with the user.

Course work is standard and includes an introduction, main part, conclusion, list of sources and application.

In the Introduction the purpose and objectives based on analysis of the original data has formulated.

The main part of the explanatory notes has given the theoretical information about functional tools used in the design, as well as a list of questions to the development and an example of the program.

In the final part of the explanatory note assesses the results of development.

**Оглавление**

**Оглавление** 8

**Введение** 9

**Теоретические основы** 10

Протоколы 10

API 12

Логирование 14

**Реализация** 16

**Пример работы программы** 18

**Заключение** 19

**Список использованных источников** 20

**Введение**

На основе анализа исходных данных сформулирована цель и поставлены задачи проектирования.

Цель работы — создание программного обеспечения для синхронизации аудиозаписей на базе открытого интерфейса социальной сети. Для достижения заданной цели в работе поставлены следующие задачи:

1. Изучить исходные библиотеки.

2. Изучить API социальной сети «ВКонтакте», выявить его недостатки для нашего проекта и провести анализ возможных решений.

3. Реализовать выбранное решение.

**Теоретические основы**

Бот – компьютерная программа, предназначенная для выполнения или имитации действий в автоматическом режиме без участия человека.

Основной задачей ботов является выполнения работы, однообразной и повторяемой, с максимально возможной скоростью, скоростью, которая очевидно намного превышает возможности человека.

Бот обычно использует тот же интерфейс, что предназначен для человека, или программы-клиента. При работе с сайтами они обмениваются по протоколу HTTP, в точности, как это происходит с браузером. IRC-, ICQ- и прочие боты в сетях общения обычно представлены так же, как обычный пользователь, и с точки зрения сервера ничем от обычных пользователей не отличаются. Использование ботов во многих случаях определяется отсутствием специального интерфейса для автоматического доступа.

В курсовой работе происходит создание бота, задачей которого является синхронизация аудиозаписей пользователя и сохранение их в заданную папку на ПК. Несмотря на широкий функционал ВКонтакте, в данной социальной сети нет возможности скачивания аудио для дальнейшего прослушивания музыки в офлайн-режиме. С данной задачей способен справиться создаваемый бот, поэтому его использование выглядит разумным решением.

Для работы бота используется всемирная сеть Интернет, а это означает, что он использует сетевые протоколы HTTP, TCP, IP, FTP. Данные протоколы находятся на разных уровнях в сетевой модели OSI и используются для различных целей.

**Протоколы**

**HTTP** - широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

Аббревиатура HTTP расшифровывается как HyperText Transfer Protocol, «протокол передачи гипертекста». В соответствии со спецификацией OSI, HTTP является протоколом прикладного (верхнего, 7-го) уровня. Актуальная на данный момент версия протокола, HTTP 1.1.

Протокол HTTP предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту. После этого клиентское приложение может продолжить отправлять другие запросы, которые будут обработаны аналогичным образом.

Задача, которая традиционно решается с помощью протокола HTTP — обмен данными между пользовательским приложением, осуществляющим доступ к веб-ресурсам (обычно это веб-браузер) и веб-сервером. На данный момент именно благодаря протоколу HTTP обеспечивается работа Всемирной паутины.

Также HTTP часто используется как протокол передачи информации для других протоколов прикладного уровня, таких как SOAP, XML-RPC и WebDAV. В таком случае говорят, что протокол HTTP используется как «транспорт».

**TCP** (англ. transmission control protocol — протокол управления передачей) — один из основных протоколов передачи данных интернета, предназначенный для управления передачей данных. Сети и подсети, в которых совместно используются протоколы TCP и IP называются сетями TCP/IP.

В стеке протоколов IP TCP выполняет функции протокола транспортного уровня модели OSI.

Механизм TCP предоставляет поток данных с предварительной установкой соединения, осуществляет повторный запрос данных в случае потери данных и устраняет дублирование при получении двух копий одного пакета, гарантируя тем самым, в отличие от UDP, целостность передаваемых данных и уведомление отправителя о результатах передачи.

Когда осуществляется передача от компьютера к компьютеру через Интернет, TCP работает на верхнем уровне между двумя конечными системами, например, браузером и веб-сервером. TCP осуществляет надежную передачу потока байтов от одной программы на некотором компьютере к другой программе на другом компьютере (например, программы для электронной почты, для обмена файлами). TCP контролирует длину сообщения, скорость обмена сообщениями, сетевой трафик.

**Internet Protocol** (IP, досл. «межсетевой протокол») — маршрутизируемый протокол сетевого уровня стека TCP/IP. Именно IP стал тем протоколом, который объединил отдельные компьютерные сети во всемирную сеть Интернет. Неотъемлемой частью протокола является адресация сети.

IP объединяет сегменты сети в единую сеть, обеспечивая доставку пакетов данных между любыми узлами сети через произвольное число промежуточных узлов (маршрутизаторов). Он классифицируется как протокол третьего уровня по сетевой модели OSI. IP не гарантирует надёжной доставки пакета до адресата — в частности, пакеты могут прийти не в том порядке, в котором были отправлены, продублироваться (приходят две копии одного пакета), оказаться повреждёнными (обычно повреждённые пакеты уничтожаются) или не прийти вовсе. Гарантию безошибочной доставки пакетов дают некоторые протоколы более высокого уровня — транспортного уровня сетевой модели OSI, — например, TCP, которые используют IP в качестве транспорта.

В современной сети Интернет используется IP четвёртой версии, также известный как IPv4. В протоколе IP этой версии каждому узлу сети ставится в соответствие IP-адрес длиной 4 октета (4 байта). При этом компьютеры в подсетях объединяются общими начальными битами адреса. В настоящее время вводится в эксплуатацию шестая версия протокола — IPv6, которая позволяет адресовать значительно большее количество узлов, чем IPv4. Эта версия отличается повышенной разрядностью адреса, встроенной возможностью шифрования и некоторыми другими особенностями.

**FTP** (англ. File Transfer Protocol — протокол передачи файлов) — стандартный протокол, предназначенный для передачи файлов по TCP-сетям (например, Интернет). Использует 21-й порт. FTP часто используется для загрузки сетевых страниц и других документов с частного устройства разработки на открытые сервера хостинга.

Протокол построен на архитектуре «клиент-сервер» и использует разные сетевые соединения для передачи команд и данных между клиентом и сервером. Пользователи FTP могут пройти аутентификацию, передавая логин и пароль открытым текстом, или же, если это разрешено на сервере, они могут подключиться анонимно. Можно использовать протокол SSH для безопасной передачи, скрывающей (шифрующей) логин и пароль, а также шифрующей содержимое.

Первые клиентские FTP-приложения были интерактивными инструментами командной строки, реализующими стандартные команды и синтаксис. Графические пользовательские интерфейсы с тех пор были разработаны для многих используемых по сей день операционных систем. Среди этих интерфейсов как программы общего веб-дизайна вроде Microsoft Expression Web, так и специализированные FTP-клиенты (например, FileZilla).

**API**

API (Application Programming Interface) — это интерфейс программирования, интерфейс создания приложений. Если говорить более понятным языком, то API — это готовый код для упрощения жизни программисту. API создавался для того, чтобы программист реально мог облегчить задачу написания того или иного приложения благодаря использованию готового кода (например, функций). Всем известный jQuery, написанный на JavaScript является тоже своего рода API. Если рассматривать конкретно данный пример, то jQuery позволяет намного облегчить написание кода. То, что обычными средствами JavaScript можно было сделать за 30 строк, через jQuery пишется через 5-6.

API определяет функциональность, которую предоставляет программа (модуль, библиотека), при этом API позволяет абстрагироваться от того, как именно эта функциональность реализована.

Если программу (модуль, библиотеку) рассматривать как чёрный ящик, то API — это множество «ручек», которые доступны пользователю данного ящика и которые он может вертеть и дёргать.

Программные компоненты взаимодействуют друг с другом посредством API. При этом обычно компоненты образуют иерархию — высокоуровневые компоненты используют API низкоуровневых компонентов, а те, в свою очередь, используют API ещё более низкоуровневых компонентов.

API принимает запросы по методам (GET & POST) и отдаёт не обычный HTML для браузеров, а результат запроса в определённом формате (XML, JSON и др.) Предназначен он скрипту со стороннего сайта/сервиса/программы, который посылает эти GET/POST запросы, получает результат и как-то использует данные.

API «ВКонтакте» позволяет пользоваться почти всеми функциями данной социальной сети, API необходим для возможного интегрирования в какие-либо программные продукты, многие почтовые ящики, интернет магазины и т.д. интегрируются в друг друга, что создает удобство пользователям. POST – запрос на создание, GET – запрос на получение, DELETE – запрос на удаление.

Работу API «ВКонтакте» мы можем проверить на примере метода audio.get, который возвращает список аудиозаписей пользователя или сообщества. Причём получаемая информация зависит от параметров запроса. Используя данный метод мы указываем следующие параметры: идентификатор владельца аудиозаписей (owner\_id), идентификатор альбома с аудиозаписями (album\_id), идентификаторы аудиозаписей, информацию о которых необходимо вернуть (audio\_ids), возврат информации о пользователях, загрузивших аудиозапись (need\_user, 1 или 0), смещение, необходимое для выборки определенного количества аудиозаписей (offset), количество возвращаемых аудиозаписей (count, максимум – 6000).

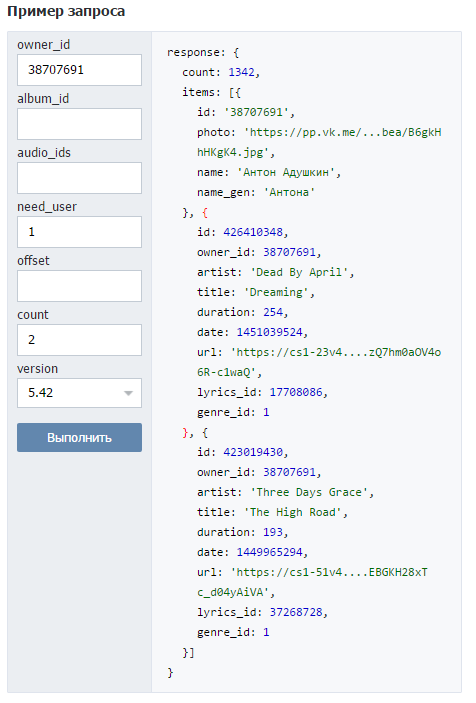


Рис.1. Метод audio.get

**Логирование**

Файл регистрации, протокол, журнал или **лог** (англ. log) — файл с записями о событиях в хронологическом порядке. Различают регистрацию внешних событий и протоколирование работы самой программы - источника записей (хотя часто всё записывается в единый файл).

Логирование используется для учета всех действий, что происходят в программе. Логирование позволяет находить ошибки, вести отчет о простых изменениях в программе и ее работе.

Уровни логирования:

* Trace – вывод всего подряд. На тот случай, если Debug не позволяет локализовать ошибку. В нём полезно отмечать вызовы разнообразных блокирующих и асинхронных операций.
* Debug – журналирование моментов вызова «крупных» операций. Старт/остановка потока, запрос пользователя и т.п.
* Info – разовые операции, которые повторяются крайне редко, но не регулярно.
* Warning – неожиданные параметры вызова, странный формат запроса, использование дефолтных значений в замен не корректных.
* Error – повод для внимания разработчиков, ошибка в работе программы.
* Fatal – вывод ошибки из-за которой работа программы невозможна.

**Реализация**

Основной задачей для бота являлась синхронизация определённого количества аудиозаписей с последующим сохранением их в заданную папку на персональном компьютере пользователя. Папка будет называться « .music » и будет располагаться в корневой папке пользователя ПК на ОС Windows.



Рис. 2. Создание объекта класса String с указанием папки для синхронизации аудио

Далее бот должен создавать эту директорию и загружать в неё аудиозаписи пользователя, предварительно высветив информацию о количестве синхронизируемой музыки.

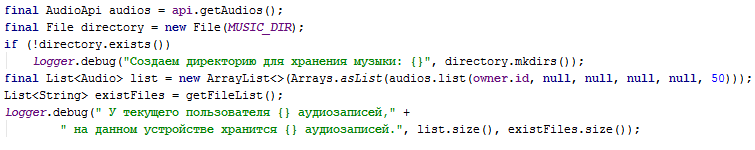


Рис. 3. Создание директории

Данный участок кода означает, что бот загрузит 50 последних аудиозаписей вошедшего в систему пользователя (owner\_id). Если последний параметр изменить на *null*, то бот загрузит все аудиозаписи пользователя.

Если синхронизация аудио проводится не в первый раз, то бот должен загрузить в директорию « .music » аудиозаписи, которые раньше не синхронизировались, и удалить “лишние” файлы, которые были удалены пользователем из своей аудиотеки после предыдущей синхронизации. В итоге мы получим, что в заданной папке будет находится актуальная синхронизированная музыка.

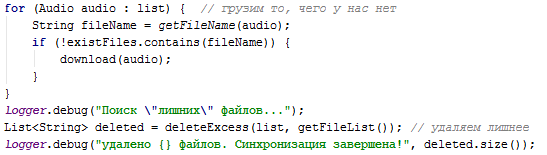


Рис. 4. Загрузка недостающего и удаление лишнего

Также следует обратить внимание на то, что в названии аудиозаписей «ВКонтакте» могут содержаться символы, используемые в синтаксисе языка Java, а также символы, которые бот при работе будет неправильно выводить на экран. Так, например, если в названии исполнителя или песни присутствует амперсанд (&), то программа по умолчанию будет выводить его как *“&amp;”*. Эта проблема решается с помощью метода replaceAll, который заменяет подобного рода символы на правильные, или просто пропускает их.

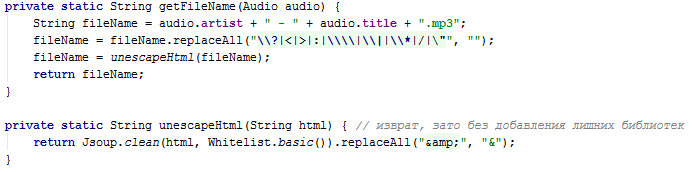


Рис. 5. Решение проблемы с нечитаемыми символами

Если в ходе работы программы не была обнаружена директория, в которую будет сохраняться музыка, то бот будет выдавать ошибку. Если же данной проблемы не возникло, то программа продолжит работу.

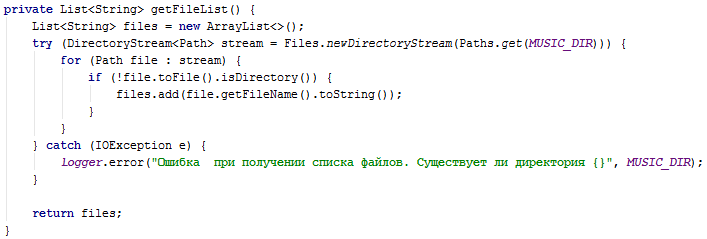


Рис. 6. Проверка существования директории

**Пример работы программы**

****

Рис. 7. Загрузка последних 20 аудиозаписей пользователя Адушкин Антон (id38707691, owner\_id)

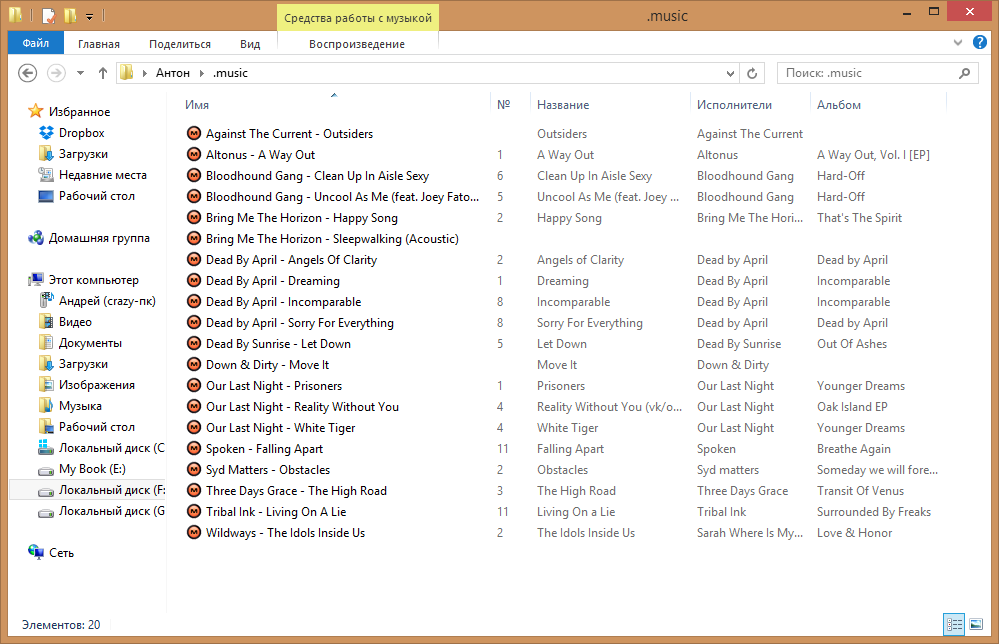


Рис. 8. Результат работы программы

**Заключение**

В ходе выполнения курсовой работы была достигнута цель, суть которой заключается в создании программного обеспечения, соответствующим исходным требованиям. В ходе разработки были изучены сетевые протоколы, используемые в программе, проанализированы возможности API ВКонтакте.

В результате работы, созданное приложение выполняет поставленную перед ней задачу синхронизации аудиозаписей.

**Список использованных источников**

1.<http://www.wikiznanie.ru/ru-wz/index.php/%D0%91%D0%BE%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)>

2. <http://habrahabr.ru/post/215117/>

3. <https://ru.wikipedia.org/wiki/TCP>

4. <https://ru.wikipedia.org/wiki/IP>

5. <https://ru.wikipedia.org/wiki/FTP>

6. <http://habrahabr.ru/sandbox/52599/>

7. <https://ru.wikipedia.org/wiki/API>

8. <http://habrahabr.ru/post/135242/>