|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
|  | | | |
|  | |  |  |
| 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»** | |
| **Тема курсовой работы: «Компьютерная модель интерактивного взаимодействия с сервисом математических расчетов «WolframAlpha» на базе открытого интерфейса социальной сети «ВКонтакте».** | |
| Студент группы ИСБО-01-13 | *Суконкин М.В.* |
| Руководитель курсовой работы | *Хлебников А. А.* |
| Рецензент | *Хлебников А. А.* |
|  |  |

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

Москва 2015

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | |  |
|  | | | | | |
| МИНОБРНАУКИ РОССИИ | | | | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего профессионального образования  **"Московский государственный информационных технологий,**  **радиотехники и электроники"**  **МИРЭА** | | | | | |
| Институт информационных технологий (ИТ) | | | | | |
| Кафедра информатики и информационных систем (ИИС) | | | | | |
|  | | | **Утверждаю** | | |
|  | | |  | | |
|  | | | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_2015 г. | | |
| **ЗАДАНИЕ** | | | | | |
| **на выполнение курсовой работы** | | | | | |
| **по** **дисциплине** «Объектно-ориентированное программирование» | | | | | |

Студент Суконкин Максим Владимирович Группа ИСБО-01-13

1. **Тема** «Компьютерная модель интерактивного взаимодействия с сервисом математических расчетов «WolframAlpha» на базе открытого интерфейса социальной сети «ВКонтакте»».
2. **Исходные данные:**

Документация API сервиса «WolframAlpha»;

Документация API социальной сети «ВКонтакте»;

JDK 1.8;

Сторонние библиотеки: okhttp, retrofit, gson, и другие.

* 1. **Перечень вопросов, подлежащих разработке, и обязательного графического материала:**

Разработать модуль расширения на базе Java Extensible Applications;

Разработать окружение тестирования;

Разработать базовый класс для первичной обработки сообщений;

Разработать классы, взаимодействующие с сервисом «WolframAlpha» посредством открытого API;

Разработать классы для обработки ответов сервиса «WolframAlpha» и формирования на их основе графического файла.

|  |  |  |  |
| --- | --- | --- | --- |
| 1. **Срок представления к защите курсовой работы:** **до** «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_\_ г. | | | |
|  | | | |
| Задание на курсовую работу выдал | «\_\_\_»\_\_\_\_\_\_\_ 201\_\_г. |  | *Хлебников А. А.* |
| Задание на курсовую работу получил | «\_\_\_»\_\_\_\_\_\_\_ 201\_\_г. |  | *Суконкин М.В.* |

УДК 004.9

ББК 32.972

*Руководитель и рецензент:* Хлебников Андрей Александрович

*Студент:* Суконкин Максим Владимирович

*Тема:* **«**Компьютерная модель интерактивного взаимодействия с сервисом математических расчетов «WolframAlpha» на базе открытого интерфейса социальной сети «ВКонтакте**»**

В ходе реализации задания на курсовую работу осуществляется исследование вопросов разработки программного обеспечения на языке программирования Java, изучение документации для работы с открытыми программными интерфейсами, написание программного кода.

Было создано соответствующее теме работы программное обеспечение и разработана документация работы, включающая в себя: утвержденное задание на курсовую работу; расчетно-пояснительную записку (РПЗ); рецензия (первичная) руководителя работы с итоговой отметкой о допуске работы к защите.

During the implementation of the task for course work done study on software development in the programming language Java, the study of the documentation for open APIs, writing software code.

It was established to work theme software and documentation developed by the work, which includes approved task for course work; settlement and explanatory note (RPGs); Reviews (primary) of the head with a final mark for admission to the work of the defense.

**Ключевые слова:** Java, JDK, ВКонтакте, WolframAlpha.

Координаты автора:

МИРЭА: 119454, Москва, пр-т Вернадского, д. 78

Кафедра Информатики и Информационных систем (ИИС)

**Реферат**

Данная курсовая работа содержит 26 страниц, 3 иллюстрации и 11 использованных источников.

Целью данной курсовой работы является разработка программного обеспечения на языке программирования Java.

Основными задачами явились: изучение документации Java, API «ВКонтакте», API «WolframAlpha», сторонних библиотек и протоколов, изучение среды разработки, создание программных модулей и классов, написание программного кода, реализующего необходимый функционал.

The purpose of this course is to develop software in the programming language Java.

The main objectives were: to study the documentation Java, API «VKontakte», API «WolframAlpha», third-party libraries and protocols, the study of the development environment, the creation of program modules and classes, writing code that implements the required functionality.

Оглавление

[**Введение** 6](#_Toc438243879)

[**Глава 1. Теоретическая часть** 7](#_Toc438243880)

[**1.1. Составление технического задания по реализации курсовой работы** 7](#_Toc438243881)

[**1.2. Описание сервиса «WolframAlpha»** 8](#_Toc438243882)

[**1.3. Описание открытого программного интерфейса «ВКонтакте»** 9](#_Toc438243883)

[**1.4. Описание открытого программного интерфейса «WolframAlpha»** 14](#_Toc438243884)

[**1.5. Описание протокола HTTP** 14](#_Toc438243885)

[**1.6. Некоторые используемые сторонние библиотеки** 18](#_Toc438243886)

[**Глава 2. Практическая часть** 19](#_Toc438243887)

[**2.1. Разработка архитектуры курсовой работы** 19](#_Toc438243888)

[**2.2. Разработка модуля взаимодействия с API «WolframAlpha»** 20](#_Toc438243889)

[**2.3. Сборка проекта и проверка работоспособности программы** 22](#_Toc438243890)

[**Заключение** 25](#_Toc438243891)

[**Список использованной литературы** 25](#_Toc438243892)

**Введение**

Согласно теме курсовой работы, необходимо разработать компьютерную модель интерактивного взаимодействия с сервисом математических расчетов «WolframAlpha» на базе открытого интерфейса социальной сети «ВКонтакте».

Курсовая работы выполнена на языке программирования Java (JDK 1.8), с применением целого ряда сторонних библиотек и модулей. В качестве среды разработке использована программа IntelliJ IDEA. Для контроля версий файлов применяется система контроля версий Git.

В результате выполнения курсовой работы ожидается получение готовой к работе программы, соответствующей теме работы.

При разработке программного обеспечения с подобным функционалом неизбежно обращение к сторонним сервисам. В данной работе используются сайты «ВКонтакте» и «WolframAlpha».

Обмен данными между локальным компьютером, на котором выполняется программа, и удаленным сервером, ведется при помощи API. API (интерфейс программирования приложений, интерфейс прикладного программирования) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах.

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

По такому принципу построены протоколы передачи данных по Интернет. Стандартный стек протоколов (сетевая модель OSI) содержит 7 уровней (от физического уровня передачи бит до уровня протоколов приложений, подобных протоколам HTTP и IMAP). Каждый уровень пользуется функциональностью предыдущего («нижележащего») уровня передачи данных и, в свою очередь, предоставляет нужную функциональность следующему («вышележащему») уровню. Понятие протокола близко по смыслу к понятию API. И то, и другое является абстракцией функциональности, только в первом случае речь идёт о передаче данных, а во втором — о взаимодействии приложений.

Следует заметить, что в рамках данной курсовой работы используются веб-сервисы. Их API реализуется при помощи набора HTTP-запросов, а также через определение структуры HTTP-ответов, для выражения которых используют XML или JSON форматы.

Для реализации курсовой работы используется Java Extensible Applications, т.е. модульная система построения приложений. Каждая часть программы отделена от других по функциональному признаку и осуществляет взаимодействие с другими при помощи интерфейсов.

**Глава 1. Теоретическая часть**

* 1. **Составление технического задания по реализации курсовой работы**

Перед началом написания программного кода, необходимо в общих чертах определить архитектуру программы, основные используемые библиотеки, взаимодействие между классами, модели взаимодействия между различными компонентами программы друг с другом и с внешними сервисами.

Итак, программа состоит из трех основных частей:

1. Базовый модуль. Выполняет взаимодействие с серверами социальной сети «ВКонтакте» посредством API. Включает в себя реализацию таких функций, как авторизация (с помощью пары логин-пароль либо по токену), поддержка соединения, отправка запросов к внешним серверам и получение ответа, разбор этого ответа. Также включает в себя реализацию множества дополнительных утилитарных функций.
2. Модуль взаимодействия с сервисом «WolframAlpha». Включает в себя несколько классов, каждый из которых предназначен для выполнения своей задачи. Необходимо реализовать классы для взаимодействия с базовым модулем, для разбора входящих сообщений и отправки исходящих, а также для отправки запросов на сервера «WolframAlpha» и разбора полученных ответов.
3. Модуль запуска, который запускается на выполнение и загружает другие модули.

Согласно документации API «WolframAlpha», ответ на запрос программа получит в виде XML-документа. Так как предполагается отправка результата в виде, понятном человеку, необходимы дополнительные библиотеки для разбора ответа сервера.

* 1. **Описание сервиса «WolframAlpha»**

Основной функционал курсового проекта заключается в интерактивном обмене информацией с сервисом «WolframAlpha».

Основанная Стивеном Вольфрамом в 1987 году, компания Wolfram Research является одной из наиболее авторитетных компаний по производству программного обеспечения в мире, а также активным инноватором в области научных и технических разработок. Будучи новатором в области компьютерных вычислениях и в предоставлении вычислительных знаний, компания воплотила в жизнь давнее намерение создать научную и техническую базу, а также необходимый инструментарий для того, чтобы сделать вычисления всё более мощной силой в сегодняшнем и завтрашнем мире.

Wolfram|Alpha — база знаний и набор вычислительных алгоритмов (англ. computational knowledge engine), вопросно-ответная система. Запущена 15 мая 2009 года. Не является поисковой системой.

Wolfram|Alpha не возвращает перечень ссылок, основанный на результатах запроса, а вычисляет ответ, основываясь на собственной базе знаний, которая содержит данные о математике, физике, астрономии, химии, биологии, медицине, истории, географии, политике, музыке, кинематографии, а также информацию об известных людях и интернет-сайтах. Он способен переводить данные между различными единицами измерения, системами счисления, подбирать общую формулу последовательности, находить возможные замкнутые формы для приближенных дробных чисел, вычислять суммы, пределы, интегралы, решать уравнения и системы уравнений, производить операции с матрицами, определять свойства чисел и геометрических фигур.

Движок Wolfram|Alpha основан на обработке естественного языка, большой библиотеке алгоритмов и NKS-подходе для ответов на запросы. Он написан на языке Mathematica и составляет около 5 миллионов строк, в настоящее время выполняется примерно на 10000 процессорах.

WolframAlpha - математический процессор онлайн, процессор знаний, который по запросу предоставляет данные об окружающем мире в числах.

Поскольку математический движок WolframAlpha работает на основе алгоритмов известной системы компьютерной математики Mathametica, этим результатам можно полностью доверять.

База знаний, из которой черпает свои способности WolframAlpha, постоянно пополняется актуальными материалами, фактическими и числовыми данными, алгоритмами - с каждым днем WolframAlpha становится "умнее". Возможности этой системы лучше всего позволяют оценить многочисленные примеры ее использования из разных областей знания.

Примеры команд, с которыми «WolframAlpha» легко справится:

* weather forecast in Moscow – показать прогноз погоды в Москве;
* 9620 to binary – конвертировать число в двоичную систему счисления;
* integrate (x^2)dx/(x^2 - 5) – вычислить интеграл;
* 3х-2х2+1=-1 – решить уравнение;
* plot x^2+1 – построить график;
* 120 meters – произвести конвертацию физической величины;
* #FE3456 – вывести информацию о цвете по его коду;
* 5, 14, 23, 32, 41, ... – продолжить числовой ряд;
* y'' + y = 0 – решить дифференциальное уравнение;

В курсовой работе реализуется интерфейс для взаимодействия пользователя с сервисом «WolframAlpha» через социальную сеть «ВКонтакте». Программа пишется таким образом, чтобы не было ограничений на множество входных и выходных команд. Это значит, что конечный пользователь сможет работать со всеми областями знаний и типами команд, которые поддерживает «WolframAlpha».

* 1. **Описание открытого программного интерфейса «ВКонтакте»**

API сайта “ВКонтакте” представляет собой удобный инструмент для разработчика, который позволяет использовать широкий функционал данной социальной сети в своих программах. Из основный функций, полезных для разработчика и предоставляемых API можно отметить такие:

* Подключение к странице пользователя
* Получение информации о пользователе
* Получение списка друзей пользователя
* Работа с группами, получение информации о группах
* Работа с изображениями, видео, музыкой пользователя
* Отправка сообщений пользователям и группам
* Работа со стеной, отправление записей на стену

Для начала работы с API необходимо зарегистрировать приложение.

Сделать это можно на странице: *https://vk.com/editapp?act=create*.

После добавления приложения будет выдан его числовой идентификатор, при помощи которого и будет осуществляться вся дальнейшая работа с API.

Для доступа к API ВКонтакте из любого Standalone-приложения предусмотрен механизм клиентской авторизации на базе протокола OAuth 2.0. В качестве клиента может выступать любое Desktop/мобильное приложение, имеющее доступ к управлению Web-браузером. В данной курсовой работе будет производится эмуляция запросов Web-браузера посредством специальных библиотек JDK.  
  
Согласно документации, процесс аутентификации приложения состоит из 3-х шагов:

1. Открытие окна браузера для аутентификации пользователя на сайте ВКонтакте.
2. Разрешение пользователем доступа к своим данным.
3. Передача в приложение ключа access\_token для доступа к API.

При клиентской авторизации ключ доступа к API access\_token выдаётся приложению без необходимости раскрытия секретного ключа приложения. Механизм клиентской авторизации имеет название Implicit Flow в соответствии со спецификацией протокола OAuth 2.0.   
После прохождения пользователем процедуры авторизации, дальнейшую работу с API можно осуществлять посредствам специального ключа (access\_token). Запросы нужно будет отправлять на адрес вида: *https://api.vkontakte.ru/method/METHOD\_NAME?PARAMETERS&access\_token=ACCESS\_TOKEN*

ВКонтакте умеет возвращать результаты выполнения запросов как в JSON-формате, так и в XML. После успешной авторизации приложения браузер пользователя будет перенаправлен по адресу REDIRECT\_URI, указанному при открытии диалога авторизации. При этом ключ доступа к API access\_token и другие параметры будут передана в URL-фрагменте ссылки:

*http://REDIRECT\_URI#access\_token=533bacf01e11f55b536a565b57531ad114461ae8736d6506a3&expires\_in=86400&user\_id=8492*

Вместе с ключом access\_token также будет указано время его жизни expires\_in, заданное в секундах. Если срок использования ключа истек, то необходимо повторно провести все описанные выше шаги, но в этом случае пользователю уже не придется дважды разрешать доступ. Запрашивать access\_token также необходимо при смене пользователем логина или пароля или удалением приложения в настройках доступа.

Кроме того, среди возвращаемых параметров будет указан user\_id - идентификатор авторизовавшегося пользователя в социальной сети.

В случае возникновения ошибки авторизации в качестве GET-параметров в REDIRECT\_URI будет передана информация об этой ошибке.

*http://REDIRECT\_URI?error=access\_denied&error\_description=The+user+or+authorization+server+denied+the+request.*

Для того чтобы вызвать метод API ВКонтакте, Вам необходимо осуществить POST или GET запрос по протоколу HTTPS на указанный URL:

*https://api.vk.com/method/METHOD\_NAME?PARAMETERS&access\_token=ACCESS\_TOKEN*

METHOD\_NAME – название метода из списка функций API,

PARAMETERS – параметры соответствующего метода API,

ACCESS\_TOKEN – ключ доступа, полученный в результате успешной авторизации приложения.

Пример:

*https://api.vk.com/method/getProfiles?uid=66748&access\_token=533bacf01e11f55b536a565b57531ac114461ae8736d6506a3*

В ответ на такой запрос сервер вернет ответ в формате JSON.

Параметры могут передаваться как методом GET, так и POST. Если планируется передавать большие данные (больше 2 килобайт), то следует использовать метод POST.

Каждый метод имеет собственный набор поддерживаемых параметров, однако существуют параметры, которые принимают все методы:

*lang* — определяет язык, на котором будут возвращаться различные данные, например названия стран и городов. Также если указан не русский, украинский и белорусский язык, то кириллические имена будут автоматически транслированы в латиницу. ru - Русский, ua - Украинский, be - Белорусский, en- Английский, es - Испанский, fi - Финский, de - Немецкий, it - Итальянский. Также поддерживается числовой идентификатор языка, возвращаемый методом account.getInfo.

*v* — обозначает версию API, которая используется. Использование этого параметра применяет некоторые изменения в формате ответа различных методов. На текущий момент актуальная версия API — 5.41. Данный параметр следует передавать со всеми запросами к API.

*https* – 1 — позволяет получать https ссылки на фотографии и другое медиа. 0 - методы возвращают http ссылки (по умолчанию). Обратите внимание, что этот параметр работает только если запросы к API осуществляются через https.

*test\_mode*=1 — тестовый режим, позволяет выполнять запросы из приложения без его включения на всех пользователей.

Со стороны клиента можно обращаться к методам API не чаще 3 раз в секунду.

Максимальное число запросов со стороны сервера зависит от числа пользователей, установивших приложение.

Если приложение установило меньше 10 000 человек, то можно совершать 5 запросов в секунду, до 100 000 – 8 запросов, до 1 000 000 – 20 запросов, больше 1 млн. – 35 запросов в секунду.

При превышении этих значений сервер вернет ошибку с кодом 6: "Too many requests per second.".

Если логика приложения подразумевает вызов нескольких методов подряд, имеет смысл обратить внимание на метод execute.

Помимо ограничений на частоту обращений, существуют и количественные ограничения на вызов однотипных методов. По понятным причинам, мы не предоставляем информацию о точных лимитах.

После превышения количественного лимита доступ к конкретному методу может требовать ввода капчи (captcha\_error), а затем может быть временно ограничен (в таком случае сервер не возвращает ответ на вызов конкретного метода, но без проблем обрабатывает любые другие запросы).

Исходя из специфики протокола HTTP, приложение, использую стандартные GET или POST запросы не может устанавливать долговременное соединение с сервером, а значит и своевременно получать от него обновления (например, о приходе нового сообщения).

Поэтому в курсовой работе необходимо использовать так называемые Long Poll сервера.

Другое название способа - "Очередь ожидающих запросов". Краткая схема такова:

1. Отправляется запрос на сервер
2. Соединение не закрывается сервером
   * пока не появится событие
3. Событие отправляется в ответ на запрос
4. Клиент тут же отправляет новый ожидающий запрос



Рис. 1. Принцип Long Poll запросов.

Каждый пакет данных, таким образом, означает новое (не учитывая Keep-Alive) соединение, которое будет открыто столько, сколько нужно, пока сервер не решит прислать информацию.

На практике, соединение обычно переустанавливается раз в 20-30 секунд, чтобы избежать возможных проблем, например, с HTTP-прокси.

В отличие от простого поллинга, здесь уведомление о событии приходит гораздо быстрее.

Задержка = установление соединения + передача данных

Такие задержки вполне терпимы в случае, если событий немного, и совершенно незаметны, если обновления с сервера приходят раз в минуту и реже.

Для того, чтобы подключиться к Long Poll серверу, необходимы следующие данные, которые можно получить, используя метод messages.getLongPollServer:

* key - секретный ключ сессии
* server - адрес сервера, к которому нужно отправлять запрос
* ts - номер последнего события, начиная с которого начинается получение данных
* mode - параметр, определяющий наличие поля прикреплений в получаемых данных с помощью битовой маски.
* Cумма номеров необходимых опций: 2 - получать прикрепления, 8 - возвращать расширенный набор событий, 32 - возвращать pts, для работы метода messages.getLongPollHistory без ограничения в 256 последних событий, 64 - в событии с кодом 8 (друг стал онлайн) возвращать в третьем поле дополнительные данные.

Для подключения нужно составить запрос следующего вида:

*http://{$server}?act=a\_check&key={$key}&ts={$ts}&wait=25&mode=2*

Где вместо {$...} - соответствующий параметр, полученный методом messages.getLongPollServer.

Принцип работы Long Poll соединения заключается в том, что сервер, получив запрос, удерживает его до тех пор, пока не произойдёт событие или не истечёт время, указанное в параметре wait.

Как только клиент получает ответ, он может послать новый запрос, указав в параметре ts новое значение, полученное в ответе, для получения новых событий.

Long Poll поддерживает передачу сообщений только в формате JSON.

* 1. **Описание открытого программного интерфейса «WolframAlpha»**

Для доступа к API «WolframAlpha» необходимо получить идентификатор приложения (AppID). Для этого необходимо пройти регистрацию разработчика на сайте системы «WolframAlpha». AppID позволяет производить до 2000 запросов в месяц в автоматическом режиме.

После получения идентификатора, для совершения запроса достаточно отправить HTTP GET запрос по адресу вида http://api.wolframalpha.com/v2/query?input=XXXX&appid=YYYY, где XXXX – входной текст, а YYYY – идентификатор приложения.

В ответ на этот запрос сервер «WolframAlpha» вернет XML-документ в кодировке UTF-8, содержащий всю доступную информацию и результаты вычислений. XML-документ может иметь различную структуру в зависимости от исходного запроса.

* 1. **Описание протокола HTTP**

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, описана в спецификации RFC 2616.

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».

API многих программных продуктов также подразумевает использование HTTP для передачи данных — сами данные при этом могут иметь любой формат, например, XML или JSON.

Как правило, передача данных по протоколу HTTP осуществляется через TCP/IP-соединения. Серверное программное обеспечение при этом обычно использует TCP-порт 80 (и, если порт не указан явно, то обычно клиентское программное обеспечение по умолчанию использует именно 80-й порт для открываемых HTTP-соединений), хотя может использовать и любой другой.

Центральным объектом в HTTP является ресурс, на который указывает URL в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д. Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя изначально данный протокол предназначен для передачи символьной информации. На первый взгляд это может показаться излишней тратой ресурсов. Действительно, данные в символьном виде занимают больше памяти, сообщения создают дополнительную нагрузку на каналы связи, однако подобный формат имеет много преимуществ. Сообщения, передаваемые по сети, удобочитаемы, и, проанализировав полученные данные, системный администратор может легко найти ошибку и устранить ее. При необходимости роль одного из взаимодействующих приложений может выполнять человек, вручную вводя сообщения в требуемом формате.

В отличие от многих других протоколов, HTTP является протоколом без памяти. Это означает, что протокол не хранит информацию о предыдущих запросах клиентов и ответах сервера. Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами. Например, клиентское веб-приложение, посылающее запросы, может отслеживать задержки ответов, а веб-сервер может хранить IP-адреса и заголовки запросов последних клиентов.

Все программное обеспечение для работы с протоколом HTTP разделяется на три основные категории:

* Серверы - поставщики услуг хранения и обработки информации (обработка запросов).
* Клиенты - конечные потребители услуг сервера (отправка запросов).
* Прокси-серверы для поддержки работы транспортных служб.

Основными клиентами являются браузеры, например: Internet Explorer, Opera, Mozilla Firefox и другие. Наиболее популярными реализациями веб-серверов являются: Internet Information Services (IIS), Apache, lighttpd, nginx. Наиболее известные реализации прокси-серверов: Squid, UserGate, Multiproxy, Naviscope.

"Классическая" схема HTTP-сеанса выглядит так.

1. Установление TCP-соединения.
2. Запрос клиента.
3. Ответ сервера.
4. Разрыв TCP-соединения.

Таким образом, клиент посылает серверу запрос, получает от него ответ, после чего взаимодействие прекращается. Обычно запрос клиента представляет собой требование передать HTML-документ или какой-нибудь другой ресурс, а ответ сервера содержит код этого ресурса.

В состав HTTP-запроса, передаваемого клиентом серверу, входят следующие компоненты.

* Строка состояния (иногда для ее обозначения используют также термины строка-статус, или строка запроса).
* Поля заголовка.
* Пустая строка.
* Тело запроса.

Строку состояния вместе с полями заголовка иногда называют также заголовком запроса.

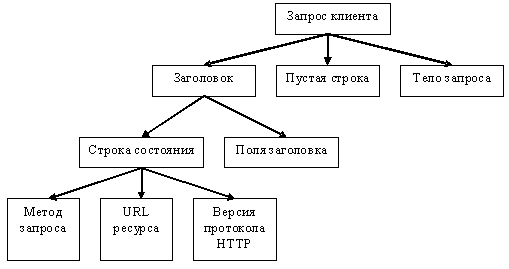


Рис. 2. Структура запроса клиента.

Строка состояния имеет следующий формат:

*метод\_запроса URL\_pecypca версия\_протокола\_НТТР*

Метод, указанный в строке состояния, определяет способ воздействия на ресурс, URL которого задан в той же строке. Метод может принимать значения GET, POST, HEAD, PUT, DELETE и т.д. Несмотря на обилие методов, для веб-программиста по-настоящему важны лишь два из них: GET и POST.

GET. Согласно формальному определению, метод GET предназначается для получения ресурса с указанным URL. Получив запрос GET, сервер должен прочитать указанный ресурс и включить код ресурса в состав ответа клиенту. Ресурс, URL которого передается в составе запроса, не обязательно должен представлять собой HTML-страницу, файл с изображением или другие данные. URL ресурса может указывать на исполняемый код программы, который, при соблюдении определенных условий, должен быть запущен на сервере. В этом случае клиенту возвращается не код программы, а данные, сгенерированные в процессе ее выполнения. Несмотря на то что, по определению, метод GET предназначен для получения информации, он может применяться и в других целях. Метод GET вполне подходит для передачи небольших фрагментов данных на сервер.

POST. Согласно тому же формальному определению, основное назначение метода POST - передача данных на сервер. Однако, подобно методу GET, метод POST может применяться по-разному и нередко используется для получения информации с сервера. Как и в случае с методом GET, URL, заданный в строке состояния, указывает на конкретный ресурс. Метод POST также может использоваться для запуска процесса.

Методы HEAD и PUT являются модификациями методов GET и POST.

Большинство протоколов предусматривают установление TCP-сессии, в ходе которой один раз происходит авторизация, и дальнейшие действия выполняются в контексте этой авторизации. HTTP же устанавливает отдельную TCP-сессию на каждый запрос; в более поздних версиях HTTP было разрешено делать несколько запросов в ходе одной TCP-сессии, но браузеры обычно запрашивают только страницу и включённые в неё объекты (картинки, каскадные стили и т. п.), а затем сразу разрывают TCP-сессию. Для поддержки авторизованного (неанонимного) доступа в HTTP используются cookies; причём такой способ авторизации позволяет сохранить сессию даже после перезагрузки клиента и сервера.

При доступе к данным по FTP или по файловым протоколам тип файла (точнее, тип содержащихся в нём данных) определяется по расширению имени файла, что не всегда удобно. HTTP перед тем, как передать сами данные, передаёт заголовок «Content-Type: тип/подтип», позволяющую клиенту однозначно определить, каким образом обрабатывать присланные данные. Это особенно важно при работе с CGI-скриптами, когда расширение имени файла указывает не на тип присылаемых клиенту данных, а на необходимость запуска данного файла на сервере и отправки клиенту результатов работы программы, записанной в этом файле (при этом один и тот же файл в зависимости от аргументов запроса и своих собственных соображений может порождать ответы разных типов — в простейшем случае картинки в разных форматах).

* 1. **Некоторые используемые сторонние библиотеки**

Библиотека OkHTTP - это HTTP клиент. Он основан на кодах HttpURLConnection и имеет ряд дополнительных функций. В частности, в OkHTTP:

* добавлена поддержка протоколов HTTP 2 (draft), SPDY 3 (draft);
* реализовано автоматическое восстановление соединения, при возникновении распространенных сетевых проблем (например, проблем c прокси-сервером и TLS рукопожатием);
* реализован пул соединений, обеспечивающий повторное использование HTTP и SPDY соединений, за счет чего увеличивается пропускная способность и снижается время ожидания.

Retrofit является REST-клиентом для удобной работы в Java. Библиотекой удобно пользоваться для запроса к различным веб-сервисам с командами GET, POST, PUT, DELETE. Может работать в асинхронном режиме, что избавляет от лишнего кода. В курсовой работе библиотека применяется для отправки запросов к серверам WolframAlpha.

Преимущества этой библиотеки:  
1) Значительно сокращает длину кода и, соответственно, ускоряет разработку  
3) Динамически строит запросы  
4) Автоматически конвертирует JSON в объекты (используется библиотека Gson)  
5) Обрабатывает ошибки  
6) Умеет передавать файлы

Вся логика работы библиотеки построена на аннотациях. Благодаря ним можно создавать динамические запросы на сервер. Описание запросов к серверу происходит в интерфейсе. Над каждым методом должна стоять аннотация, с помощью которой Retrofit “узнает”, какого типа запрос. Также с помощью аннотаций можно указывать параметры запроса.

Еще одна из применяемых в проекте библиотек – Gson. Это библиотека Java, которая может быть использована для преобразования объектов в их представлении JSON. Она также может быть использована для преобразования строки JSON в эквивалентной объект Java.

Gson обеспечивает простые методы toJson () и fromJson () для преобразования Java объектов в JSON и наоборот, а также поддерживает произвольные сложные объекты (с глубокой иерархией наследования и широкого использования общих типов).

**Глава 2. Практическая часть**

**2.1. Разработка архитектуры курсовой работы**

Первая часть приложения – основной модуль. Он содержит внутри себя различные классы, которые реализуют интерфейсы, необходимые для работы с API «ВКонтакте».

Некоторые функции этого модуля:

* Считывание из файла информации для аутентификации;
* Проведение аутентификации путем отправки необходимых запросов и анализа ответов;
* Установка и поддержка HTTP-соединения;
* Анализ сообщений от сервера.

Вторая часть приложения – модуль взаимодействия с сервисом «WolframAlpha».

Данный модуль выполняет следующие функции:

* Принимает входящее сообщение, разбирает его на части, при необходимости отправляя на дальнейшую обработку;
* Формирует запрос к серверу «WolframAlpha», считывая из файла идентификатор приложения;
* Используя библиотеку retrofit, отправляет запрос к серверу «WolframAlpha», и получает ответ в виде XML-документа;
* Анализирует полученный ответ; в случае необходимости производя детальный разбор полученного XML-документа;
* Создает на основе XML-документа изображение, «склеивая» его из различных видов информации: текстовой и графической, полученной дополнительными HTTP GET-запросами на адреса из XML-документа;
* Анализируя размеры изображения, отправляет его способом, обеспечивающим максимально удобный просмотр.

Кроме основных классов, внутри этого модуля необходимо создать папку с ресурсами, в которой будет храниться файл с идентификатором приложения для взаимодействия с сервисом «WolframAlpha».

Третья часть приложения – модуль-загрузчик, который является исполняемым и запускается сразу после запуска приложения на выполнение. Его задача – в удобном виде запустить необходимый функциональный модуль.

**2.2. Разработка модуля взаимодействия с API «WolframAlpha»**

Данный модуль несет основную функциональную часть приложения, поэтому описанию его разработки стоит уделить особое внимание.

Он состоит из следующих классов:

Класс «VkWolframInteractionService» является базовым классом модуля. Данный класс получает и «разбирает» входящее сообщение, передает его другим классам на обработку, после чего отправляет ответ пользователю. В качестве дополнительной возможности реализована возможность выхода аккаунта, с помощью которого работает модуль, в «Онлайн», а также отправки статуса «…печатает» в диалог «ВКонтакте» во время отправки запроса к серверам «WolframAlpha».

Класс «WAMessage» описывает внутреннюю структуру сообщения, которое будет отправлено пользователю после запроса к серверам «WolframAlpha».

Интерфейс «WARequest» при помощи библиотеки retrofit позволяет отправить GET-запрос на сервера WolframAlpha и получить ответ.

Класс «WARequestImpl», используя интерфейс «WARequest», непосредственно отправляет запрос на сервера WolframAlpha и производит предварительный разбор ответа на уровне HTTP-кодов.

Класс «WARequestAction», используя библиотеку retrofit, генерирует HTTP-запрос к серверам WolframAlpha. Однако перед этим он производит два важных действия:

1. Загружает идентификатор приложения из файла, чтобы далее передать его в запрос;
2. Создает новый экземпляр класса okHttpClient и устанавливает в нем время ожидания, увеличенное относительно стандартных настроек. Это необходимо для выполнения сервисом WolframAlpha некоторых вычислительных расчетов, которые занимают десятки секунд.

Для этого используется следующий фрагмент кода:

*final OkHttpClient okHttpClient = new OkHttpClient();  
okHttpClient.setReadTimeout(40, TimeUnit.SECONDS);  
okHttpClient.setConnectTimeout(40, TimeUnit.SECONDS);*

Далее создается объект класса Retrofit, которому через встроенный Builder передается экземпляр класса okHttpClient.

Класс «WAAction» «координирует» работу предыдущих классов, осуществляющих запросы, а также разбирает полученный HTTP-ответ. Этот класс содержит методы для:

* «Сборки» и отправки запроса на основе входящей строки;
* Разбора XML-документа и создания на их основе изображения;
* Загрузки изображения на сервера «ВКонтакте»;
* Анализа размеров изображения и генерирования финального сообщения для отправки в качестве ответа.

Класс «ImageBuilder» является утилитарным классом, который служит для работы с изображениями. Он содержит методы для:

* Создания первичного изображения;
* Добавления текста на изображение;
* Объединения двух изображений, при этом одно из них получается в результате HTTP GET запроса с внешнего сервера;
* Записи итогового изображения во временный файл.

**2.3. Сборка проекта и проверка работоспособности программы**

После написания основного кода классов, необходимо выполнить их сборку, используя Maven.

Apache Maven — фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM (англ. Project Object Model), являющемся подмножеством XML. Проект Maven издаётся сообществом Apache Software Foundation, где формально является частью Jakarta Project.

Maven обеспечивает декларативную, а не императивную (в отличие от средства автоматизации сборки Apache Ant) сборку проекта. В файлах описания проекта содержится его спецификация, а не отдельные команды выполнения. Все задачи по обработке файлов, описанные в спецификации, Maven выполняет посредством их обработки последовательностью встроенных и внешних плагинов. Информация для сборки проекта, поддерживаемого Apache Maven, содержится в XML-файле с названием pom.xml. При запуске Maven проверяет, содержит ли конфигурационный файл все необходимые данные и все ли данные синтаксически правильно записаны.

Минимальная конфигурация включает версию конфигурационного файла, имя проекта, его автора и версию. С помощью pom.xml конфигурируются зависимости от других проектов, индивидуальные фазы процесса построения проекта (build process), список плагинов, реализующих порядок сборки.

Проект курсовой работы поделен на несколько модулей, или подпроектов, каждый со своим собственным POM. Далее написан корневой POM, через который все модули компилируются единой командой.

Maven использует принцип Maven-архетипов (англ. Archetypes). Архетип — это инструмент шаблонов, каждый из которых определён паттерном или моделью, по аналогии с который создаются производные.

Стандартная структура каталогов — одна из реализаций принципа архитипов в Maven. Следующая структура показывает важнейшие каталоги для проекта на Java:

Корневой каталог проекта: файл pom.xml и все дальнейшие подкаталоги

* src: все исходные файлы
  + src/main: исходные файлы собственно для продукта
    - src/main/java: Java-исходный текст
    - src/main/resources: другие файлы, которые используются при компиляции или исполнении, например Properties-файлы
  + src/test: исходные файлы, необходимые для организации автоматического тестирования
    - src/test/java: JUnit-тест-задания для автоматического тестирования
* target: все создаваемые в процессе работы Maven файлы
  + target/classes: компилированные Java-классы

В файле pom.xml задаются зависимости, которые имеет управляемый с помощью Maven проект. Менеджер зависимостей основан на нескольких основных принципах:

* Репозитории. Maven ищет необходимые файлы в локальных каталогах или в локальном Maven-репозитории. Если зависимость не может быть локально разрешена, Maven подключается к указанному Maven-репозиторию в сети и копирует в локальный репозиторий. По умолчанию Maven использует Maven Central Repository, но разработчик может конфигурировать и другие публичные Maven-репозитории, такие, как Apache, Ibiblio, Codehaus или Java.Net.
* Транзитивные зависимости. Необходимые библиотеки подгружаются в проект автоматически. При разрешении конфликта версий используется принцип «ближайшей» зависимости, то есть выбирается зависимость, путь к которой через список зависимых проектов является наиболее коротким.
* Исключение зависимостей. Файл описания проекта предусматривает возможность исключить зависимость в случае обнаружения цикличности или отсутствия необходимости в определённой библиотеке.
* Поиск зависимостей. Поиск зависимостей (open-source-библиотек и модулей) ведётся по их координатам (groupId, artifactId и version). Эти координаты могут быть определены с помощью специальных поисковых машин, например, Maven search engine. Например, по поисковому признаку «pop3», поисковая машина предоставляет результат с groupId="com.sun.mail" и artifactId="pop3".
* Менеджеры репозиториев. Репозитории реализуются с помощью менеджеров репозиториев Maven (Maven Repository Manager), таких как Apache Archiva, Nexus (ранее Proximity), Artifactory, Codehaus Maven Proxy или Dead Simple Maven Proxy.

После того, как создана правильная структура каталогов и файлов, корректно написаны файлы pom.xml и настроена среда разработки, проект может быть собран, классы могут быть преобразованы в байт-код, и стартовый класс может быть выполнен. Все это происходит при помощи одной команды «Run» в IDE IntelliJ IDEA.

После запуска проекта, загружается необходимый модуль. Выполняется аутентификация «ВКонтакте», устанавливается Long Poll соединение.

Когда эти действия выполнены, можно проверить работоспособность программы, работая с ней под видом обычного пользователя. Для этого достаточно зайти на сайт «ВКонтакте» через любой веб-браузер и написать на аккаунт, под которым авторизована программа, сообщение в установленном формате.

Программа может обработать любые команды, которые принимает сервис «WolframAlpha». Однако, для того, чтобы она распознала, что конкретную команду следует отправить на решение, сообщение необходимо начинать с символов «wi». Например, можно попросить программу отправить график функции x³, отправив команду *wi plot x^3*. Уже через 5 секунд мы получает ответ с требуемой информацией.

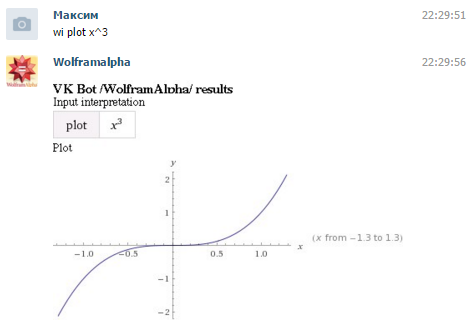


Рис. 3. Пример работы программы.

**Заключение**

Выполненная программная разработка соответствует заданию на курсовую работу. Была изучена документация открытых программных интерфейсов «ВКонтакте» и «WolframAlpha», JDK 1.8, ряда используемых сторонних библиотек и протоколов. Были разработаны несколько модулей проекта и все необходимые классы. Проведена сборка программы и проверка её работоспособности.

**Список использованной литературы**

1. Электронная энциклопедия Википедия//HTTP. URL: <https://ru.wikipedia.org/wiki/HTTP> (дата обращения - 05.11.2015)

2.Oracle Documentation// The Java Tutorials. URL: <https://docs.oracle.com/javase/tutorial/> (дата обращения - 05.11.2015)

3. TutorialsPoint // Maven Tutorial. URL: <http://www.tutorialspoint.com/maven/> (дата обращения – 06.11.2015)

4. Сообщество Habrahabr // Apache Maven — основы. URL: <http://habrahabr.ru/post/77382/> (дата обращения - 08.11.2015)

5. OkHttp//OkHttp Documentation. URL: <http://square.github.io/okhttp/> (дата обращения - 09.10.2015)

6. Square Open Source // Retrofit. URL: <http://square.github.io/retrofit/> (дата обращения - 09.11.2015)

7. Oracle// Java SE Development Kit 8 Documentation. URL: <http://www.oracle.com/technetwork/java/javase/documentation/jdk8-doc-downloads-2133158.html> (дата обращения - 10.11.2015)

8. Wolfram|Alpha // Webservice API Reference. URL: <http://products.wolframalpha.com/api/documentation.html> (дата обращения - 11.11.2015)

9. «ВКонтакте» // Документация платформы. URL: <https://vk.com/dev> (дата обращения – 12.11.2015)

10. GoogleCode //Gson Documentation. URL: <https://google-gson.googlecode.com/svn/trunk/gson/docs/javadocs/com/google/gson/Gson.html> (дата обращения – 13.11 2015)

11. Электронная энциклопедия Википедия. // WolframAlpha URL: <https://ru.wikipedia.org/wiki/WolframAlpha> (дата обращения - 14.11.2015)