Содержание

Введение

Автоматизированный сбор статистических данных и их анализ – это сильный инструмент для обработки больших объемов данных за короткое время.

В наше время системы анализа данных пользуются большой популярностью во всех сферах жизни общества, которые подвергаются неминуемой автоматизации. Одной из таких сфер является образование, поскольку оно всегда нуждается в улучшении качества процесса обучения, которое обеспечивается путем соответствия актуальности основных образовательных тем.

Таким образом, создание автоматизированной системы, помогающей пользователю получить подробную статистику в виде лайков, дизлайков, просмотров, комментариев и других статистических показателей популярности образовательных тем на самом востребованном видеохостинге YouTube поможет сэкономить уйму времени, и расширить границы аналитических возможностей образовательных тем.

Пользователями разрабатываемой информационной системы могут быть как специализированные аналитики, так и рядовые пользователи.

Целью данного курсового проекта является проектирование и разработка аналитического инструмента поиска наиболее востребованного целевой аудиторией образовательного видеоконтента по ключевым словам.

В ходе выполнения курсового проекта будут сформулированы требования к разрабатываемой системе, разработаны и декомпозированы функциональные модели IDEF0, DFD, IDEF3. Разработаны модели данных: логическая модель и физическая, также произведена разработка моделей UML. Кроме того, при работе над курсовым проектом будут реализованы некоторые функции информационной системы, которые также представлены в пояснительной записке.

1 Анализ задачи и формирование требований

Данный раздел курсовой работы предназначен для того, чтобы более точно понимать направленность решаемой задачи, а именно, разработки информационной системы аналитики роликов YouTube в интересах образовательного учреждения. Анализ задачи всегда влечёт за собой определение её требований. Первым делом необходимо произвести описание предметной области, которое поможет сформировать более чёткое понимание разрабатываемой системы.

1.1 Описание предметной области

Проектируемая система предназначена для пользователей, которые занимаются аналитикой данных в образовательном учреждении.

Аналитик данных – это специалист, который собирает, обрабатывает, изучает и интерпретирует данные. Его работа помогает принимать важные решения для улучшения качества обучения. Данное качество обеспечивается посредством соответствия актуальности основных образовательных программ к новейшим трендам в обучении, о которых можно узнать из анализа образовательных запросов на платформе YouTube. Таким образом, аналитику данных доступна функция ввода запросов в информационную систему для получения списка всех видеозаписей и статистики по каждому запросу. После получения списка видео система должна получать информацию о каждом конкретном видео из запроса, а именно информацию о названии видео, дате публикации, количестве лайков, дизлайков, комментариев и просмотров, а также получать ссылку на данное видео и embed-тег, позволяющий в будущем вставить данное видео в HTML-отчет. Доступ к данным методам должен осуществляться при наличии квоты на Google APIs аккаунте, который привязан к исполняемому на сервере скрипту.

Перед выполнением скрипта проектируемая система должна генерировать для пользователя электронный ключ. Данная функция необходима для того, чтобы обеспечить уникальность для каждого пользователя.

В разрабатываемом приложении должна иметься возможность хранения результатов анализа в базе данных, при этом необходимо генерировать новую отдельную базу данных при каждом запуске скрипта. То есть каждому успешному выполнению скрипта должна соответствовать своя отдельная база данных.

В проектируемой системе должна быть функция генерации HTML-отчета со статическими данными по популярности выбранных тем, iframe-ссылками самых востребованных видеороликов и графиками, отражающими всю полученную статистику.

После проведения анализа система должна проверять электронный ключ пользователя и перенаправлять его на страницу с соответствующим ему HTML-отчетом.

1.2 Описание требований к информационной системе

Разрабатываемая информационная система представляет собой расширение в браузере Google Chrome, которое взаимодействует с сервером, запуская на нем удаленный скрипт по анализу данных. Логика данного приложения распределена между сервером и клиентом, хранение данных осуществляется преимущественно на сервере, обмен информацией происходит по сети.

Главная задача проектируемой системы заключается в том, чтобы предоставить пользователям возможность анализировать образовательные запросы на платформе YouTube и делать выводы из данного анализа для корректировки текущих и составления новых образовательных программ, соответствующих новейшим трендам в обучении.

Основная функциональность проектируемой информационной системы представлена в предыдущем разделе. Помимо этого, для данной информационной системы свойственны следующие требования:

Системные характеристики:

- СХ-1: Система состоит из клиентской и серверной частей. Клиентская часть представлена расширением для браузера Google Chrome, написанным с помощью языка разметки HTML и языка программирования JavaScript. Серверная часть состоит из Python-скрипта, выполняющего основные бизнес-процессы.
- СХ-2: Для работы система использует интерпретатор Python.

Пользовательские требования:

- ПТ-1: Запуск анализа должен производиться после открытия расширения от браузера Google Chrome, ввода в него тематических запросов и нажатия на кнопку «Проанализировать».
- ПТ-2: По окончании работы приложение должно пересылать пользователя на HTML-отчет со всеми проанализированными данными.
- ПТ-3: Итоговый отчет должен содержать функциональность просмотра видеозаписей с самым большим количеством лайков, дизлайков, просмотров и комментариев.
- ПТ-4: Отчет должен предоставлять функциональность просмотра полученных графиков зависимостей. Переход между графиками должен быть осуществлен с помощью слайдера.

Бизнес-правила:

- БП-1: В процессе анализа каждому пользователю должен генерироваться и присваиваться свой электронный ключ пользователя, состоящий из цифр, а также латинских прописных и строчных букв длинной в 15 символов. Генерация ключа должна проводиться на стороне веб-сервера, а его значение должно передаваться в CGI-скрипт для встраивания ключа в название HTML-отчета.

- БП-2: Перед процессом получения данных необходимо проверить наличие квоты у используемых API-ключей. При наличии квоты – продолжать выполнение скрипта. При отсутствии – вывести пользователю сообщение о технических неполадках.

- БП-3: Для проведения анализа необходимо получить все необходимые данные от сервисов Google, в том числе: список всех видеозаписей по каждому запросу, а также данные о каждой конкретной видеозаписи (в том числе количество лайков, дизлайков, просмотров, комменатариев).

- БП-4: Необходимо создавать отдельную базу данных, соответствующую каждому запуску скрипта.

- БП-5: Все полученные данные от сервисов Google необходимо сохранять в созданной базе данных.

- БП-6: На основе информации, хранящейся в БД, необходимо генерировать графики, показывающие отношения количества лайков, дизлайков, просмотров, комментариев и других статистических показателей тематических запросов.

- БП-7: На основе информации, хранящейся в БД, и графиков зависимостей формировать итоговый HTML-отчет, отражающий все проанализированные данные.

Атрибуты качества:

- АК-1: Производительность

  1. Выполнение анализа не должно превышать длительность 60 секунд.

- АК-2: Устойчивость к входным данным

  1. Система должна обрабатывать входные данные пользователя с максимальной величиной до 10 запросов.

  2. Каждый запрос должен быть размером до 100 символов.

1.3 Выводы к разделу 1

В представленном разделе произведен анализ предметной области и определены основные требования, которым должна удовлетворять разрабатываемая система. Помимо этого, была сформулирована основная задача функционирования информационной системы по автоматизации аналитики роликов YouTube в интересах образовательного учреждения. Так же в данном разделе было произведено подробное описание функциональности данной ИС.

2 Структурное проектирование

Каждая информационная система должна обеспечивать требуемую производительность, функциональность, безопасность, безотказную работу, пропускную способность и множество других важнейших для эффективной работы факторов. Это достигается путем грамотного проектирования системы.

В данном разделе курсового проекта будут спроектированы функциональные модели и модели данных для информационной системы автоматизации аналитики роликов YouTube.

2.1 Разработка функциональных моделей

Функциональное моделирование является важнейшим элементом описания системы. Данный вид моделирования подразумевает описание основных бизнес-процессов в рассматриваемой предметной области. Помимо этого, построение функциональных моделей позволяет подробно описать все информационные объекты, содержащиеся в системе, назначение самой ИС и взаимосвязи с внутренними и внешними элементами.

В данном разделе будут разработаны модели IDEF0, IDEF3 и DFD. Для их построения используется векторный редактор Microsoft Visio, который предназначен для разработки диаграмм и схем.

2.1.1 Диаграмма IDEF0

Методология IDEF0 предписывает построение иерархической системы диаграмм - единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция - система разбивается на подсистемы и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности [1].

Диаграммы IDEF0 служат для представления системы как набора чередующихся функций.

Каждая диаграмма должна содержать отдельные блоки и дуги. Блоки изображают выполняемые системой функции, а дуги – связывают эти блоки и показывают типы их взаимодействия.

Диаграммы IDEF0 должны содержать не менее двух и не более семи блоков, так как это обеспечивает удобство понимания и использования данных диаграмм.

В IDEF0 различают несколько видов стрелок:

- вход – то, что перерабатывается системой;
- управление – регламентирующая и управляющая информация или правила;
- выход – результат работы системы;
- механизм – ресурсы, выполняющие работу.

Система преобразует входные потоки в выходные с учетом управления и использования механизмов.

Построение модели IDEF0 для системы автоматизации аналитики роликов YouTube в интересах образовательного учреждения начинается с построения контекстной диаграммы.

Основной функциональный блок контекстной диаграммы назовем в честь основного назначения информационной системы – анализ образовательных тем на платформе YouTube.

Входными данными контекстной диаграммы будут являться поисковые запросы, которые будет формировать непосредственно пользователь.

Управлением данной диаграммы будут являться интернет-протокол и учетная запись Google APIs, которая предоставляет возможность получения широкого спектра данных с платформы YouTube.

Выходами будут являться графики зависимостей, база данных со статистикой и HTML-отчет с проанализированными данными.

В качестве механизмов будут использоваться браузер, пользователь, информационная система и сервисы Google.

Помимо этого, необходимо указать точку зрения и цель на разрабатываемой контекстной диаграмме. Точкой зрения будет являться аналитик образовательного контента, а целью является описание процесса анализа образовательных тем на платформе YouTube.

Контекстная диаграмма будет иметь номер A-0.

Описанная выше контекстная диаграмма представлена на рисунке 2.1.1.1.

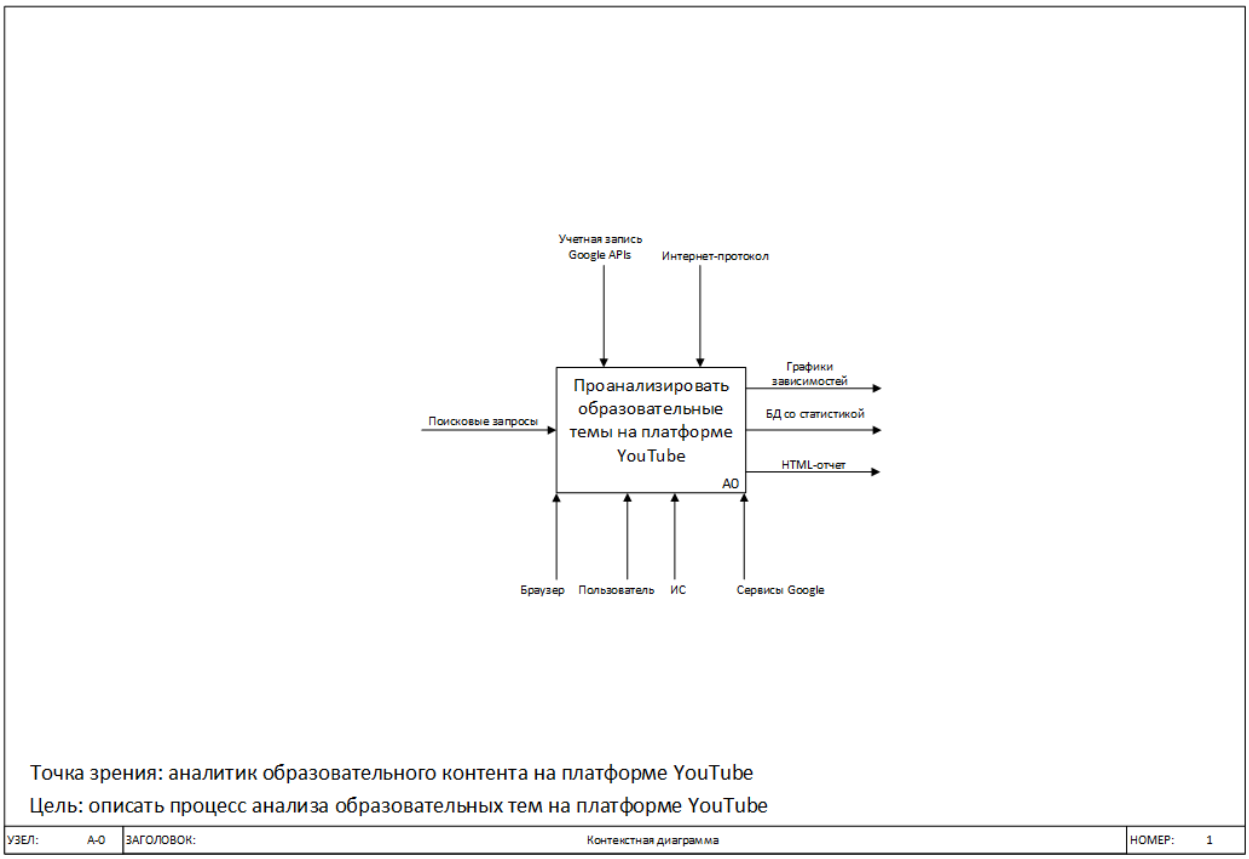

Рисунок 2.1.1.1 – Контекстная диаграмма IDEF0

На следующем этапе разработки информационной системы, необходимо декомпозировать представленную выше контекстную диаграмму.

На этапе декомпозиции выделяются основные функции разрабатываемой системы. В результате декомпозиции функция анализа образовательных тем была разбита на 4 функции:

- запустить анализ – A1;
- получить данные от сервисов Google – A2;
- сохранить полученные данные – A3;
- сформировать отчет и вывести результаты – A4.

Декомпозиция контекстной диаграммы будет иметь номер A0.

Декомпозиция контекстной диаграммы представлена на рисунке 2.1.1.2.

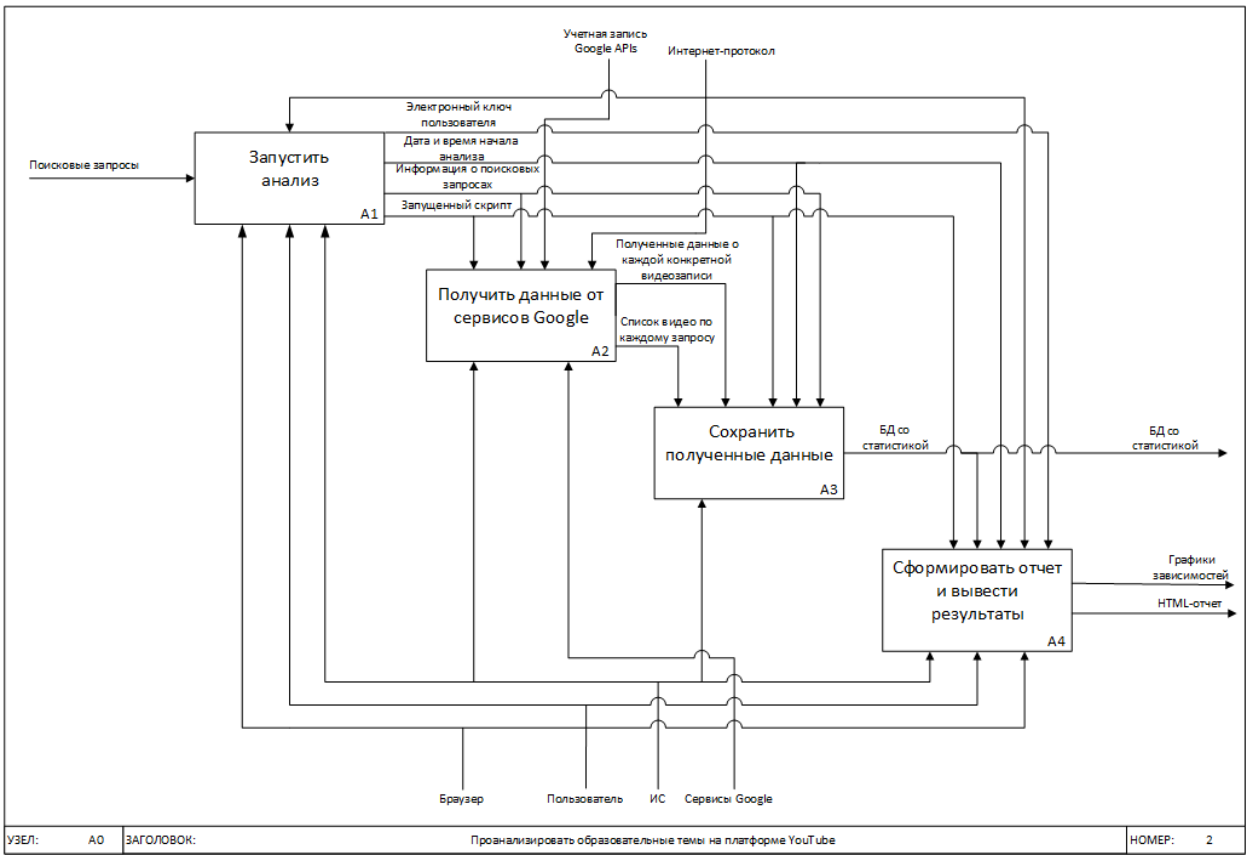

Рисунок 2.1.1.2 – Декомпозиция контекстной диаграммы IDEF0

На следующем этапе необходимо декомпозировать все функции, представленные на диаграмме A0.

Первым делом декомпозируем функцию A1, представляющую из себя запуск анализа.

Входом данной функции будут являться поисковые запросы, которые вводит пользователь.

Управлением будет выступать интернет-протокол, с помощью которого можно организовать отправку введенных данных на сервер.

Выходами данной системы будут информация о поисковых запросах, электронный ключ пользователя, дата и время начала анализа и запущенный скрипт.

В качестве механизмов будут выступать браузер, пользователь и информационная система.

Данная функция декомпозируется на блоки ввода поисковых запросов, генерации электронного ключа, а также запуска удаленного скрипта.

Декомпозиция функции A1 представлена на рисунке 2.1.1.3.

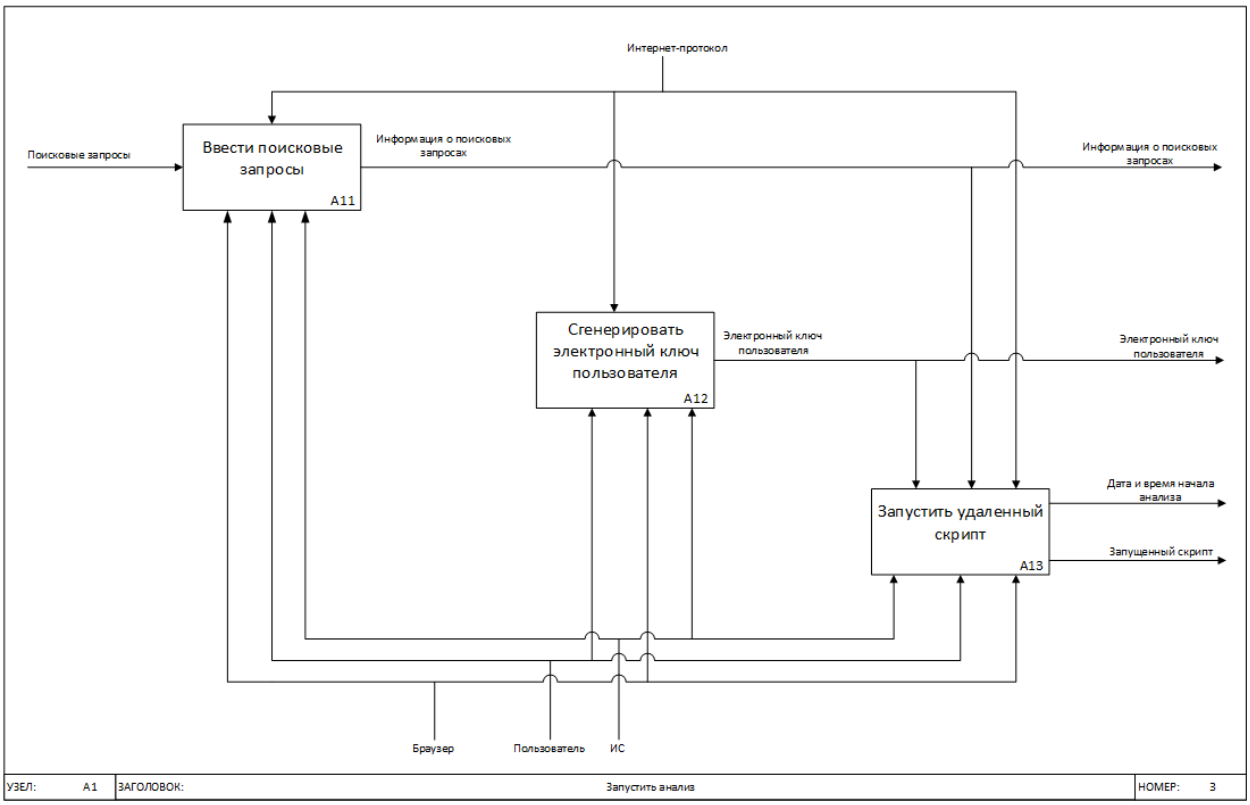

Рисунок 2.1.1.3 – Декомпозиция функции A1 диаграммы IDEF0.

После этого, необходимо декомпозировать функцию A2, представляющую из себя получение данных от сервисов Google.

Данная диаграмма не имеет входов.

Управлением будет выступать запущенный скрипт, информация о поисковых запросах, учетная запись Google APIs и интернет-протокол.

Выходами данной диаграммы будут являться список видео по каждому запросу, а также полученные данные о каждой конкретной видеозаписи.

К механизмам будут относиться браузер, пользователь и информационная система.

Диаграмма получения данных от сервисов Google будет разбита на следующие процессы: проверить квоту API-ключа, получить список всех видеозаписей по каждому запросу, получить данные о каждой конкретной видеозаписи.

Декомпозиция диаграммы получения данных от сервисов Google представлена на рисунке 2.1.1.4.

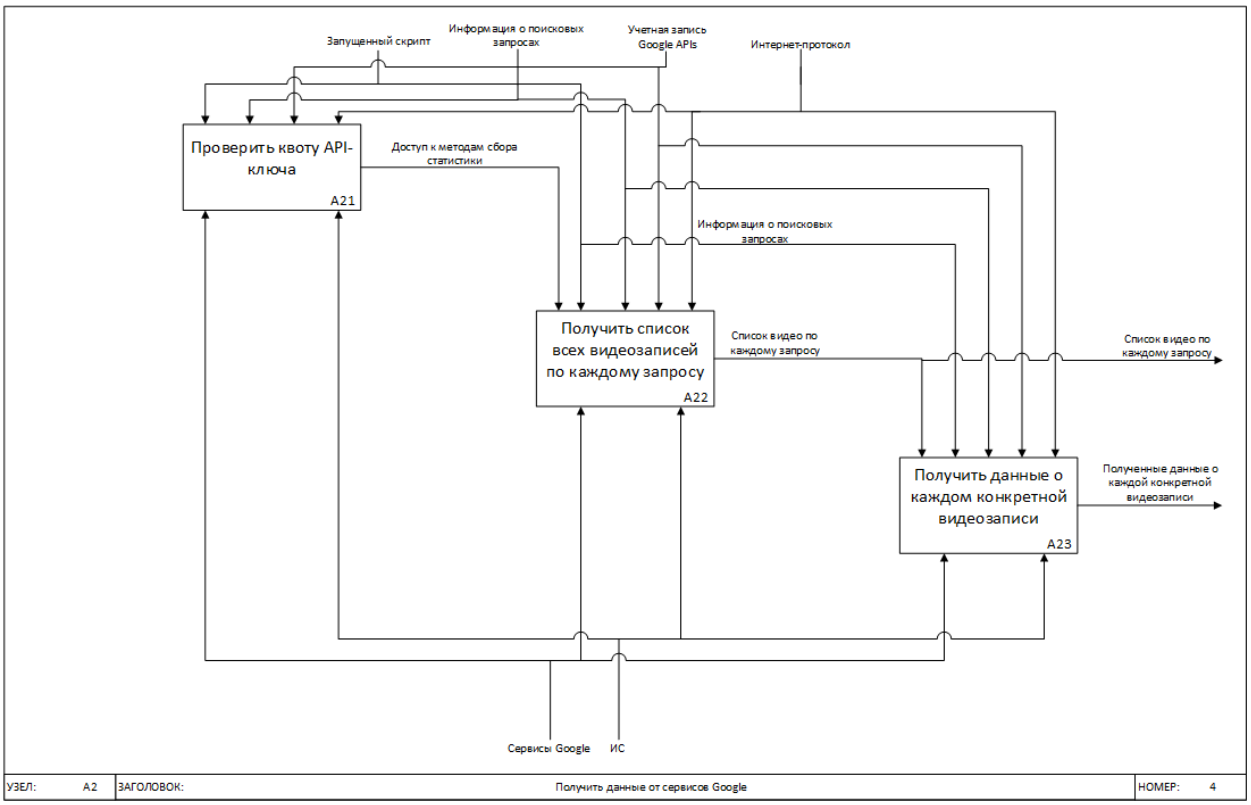

Рисунок 2.1.1.4 – Декомпозиция функции A2 диаграммы IDEF0

На следующем этапе необходимо декомпозировать функцию A3, которая представляет из себя сохранение полученных данных.

Данная диаграмма не имеет входов.

Управлением будет выступать список видео по каждому запросу, полученные данные о каждой конкретной видеозаписи, запущенный скрипт, информация о поисковых запросах, а также дата и время начала анализа.

Выходам данной диаграммы будет являться база данных со всей полученной статистикой.

К механизмам будет относиться информационная система.

Диаграмма сохранения полученных данных будет разбита на следующие процессы: создать базу данных для хранения статистики, внести в базу данных информацию о запросах, внести в базу данных информацию о списке видеозаписей, а также внести в базу данных информацию о каждой конкретной видеозаписи.

Декомпозиция диаграммы сохранения полученных данных представлена на рисунке 2.1.1.5.

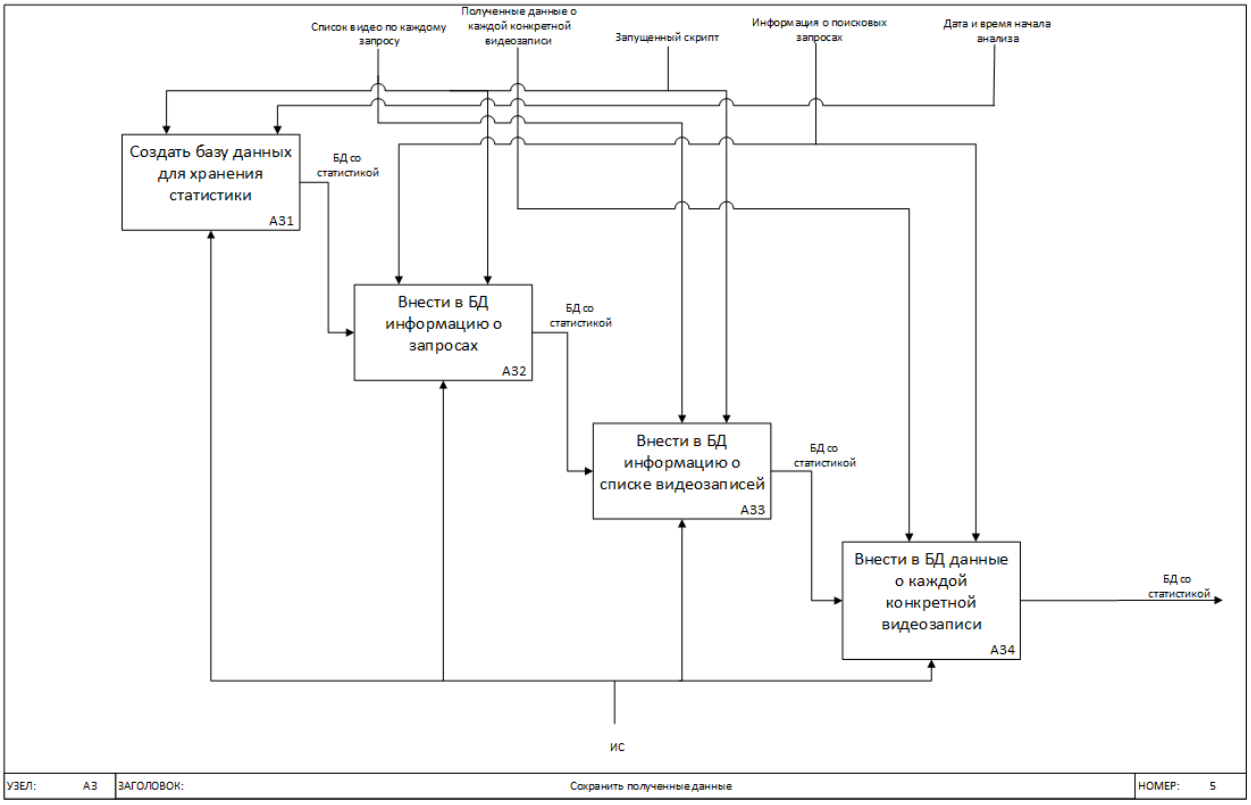

Рисунок 2.1.1.5 – Декомпозиция функции A3 диаграммы IDEF0

На следующем этапе необходимо декомпозировать функцию A4, которая представляет из себя формирование отчета и вывод результата.

Данная диаграмма не имеет входов.

Управлением будет выступать база данных со статистикой, электронный ключ пользователя, запущенный скрипт, интернет-протокол, а также дата и время начала анализа.

Выходами данной диаграммы будут графики зависимостей и HTML-отчет.

К механизмам будет относиться информационная система, пользователь и браузер.

Диаграмма формирования отчета и вывода результатов будет разбита на следующие процессы: сгенерировать графики зависимостей, сформировать HTML-отчет, открыть на клиенте HTML-отчет с соответствующим электронным ключем пользователя и датой.

Декомпозиция диаграммы формирования отчета и вывода результатов представлена на рисунке 2.1.1.6.

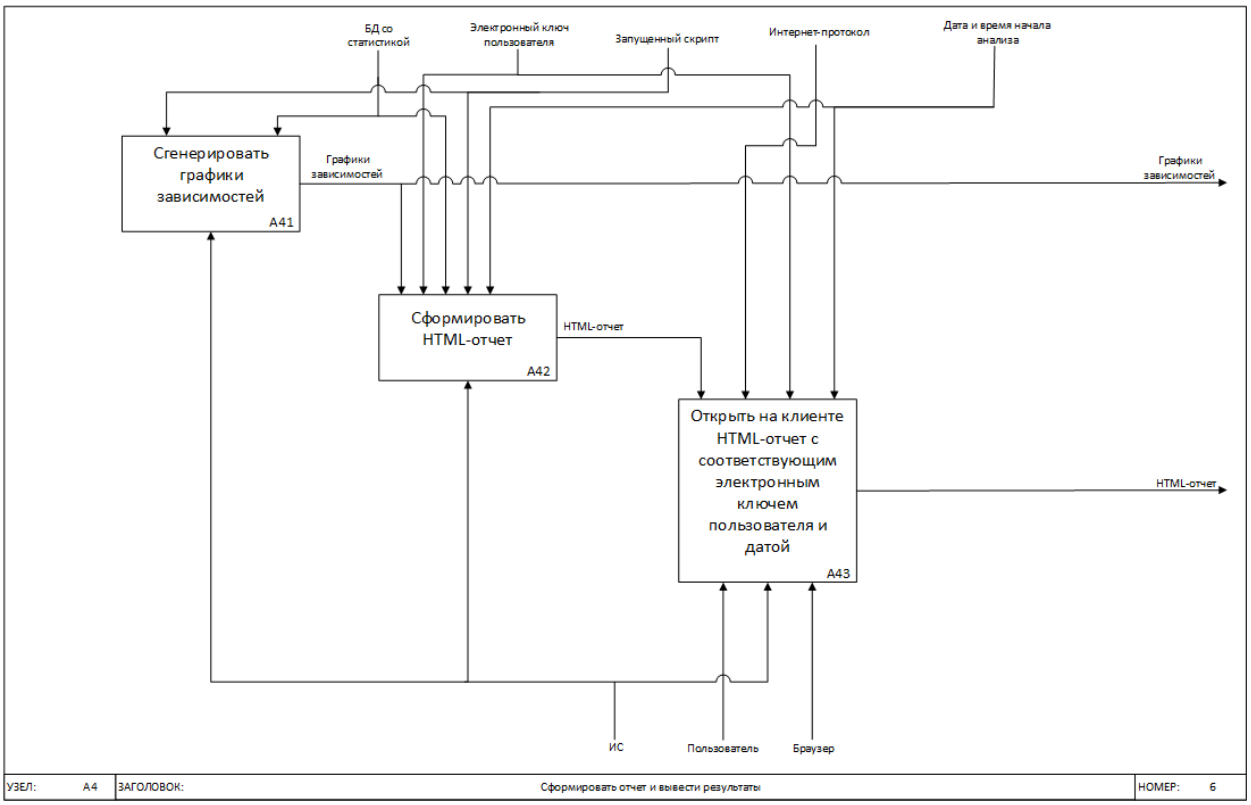

Рисунок 2.1.1.6 – Декомпозиция функции A4 диаграммы IDEF0

Таким образом, в данном подразделе курсового проекта было проведено IDEF0-проектирование разрабатываемой информационной системы.

2.1.2 Диаграмма IDEF3

IDEF3 – способ описания процессов с использованием структурированного метода, позволяющего эксперту в предметной области представить положение вещей как упорядоченную последовательность событий с одновременным описанием объектов, имеющих непосредственное отношение к процессу [2].

К основным элементам данной модели можно отнести:

- единицы работ – основные компоненты модели;
- связи – указывают взаимоотношения работ;
- перекрестки – отвечают за логику взаимодействия стрелок при их слиянии и разветвлении;
- объекты ссылок.

Основной единицей работы системы автоматизации аналитики роликов YouTube является блок «Проанализировать образовательные темы на платформе YouTube», который изображен на рисунке 2.1.2.1. Данная единица работы обозначена номером 1.1.

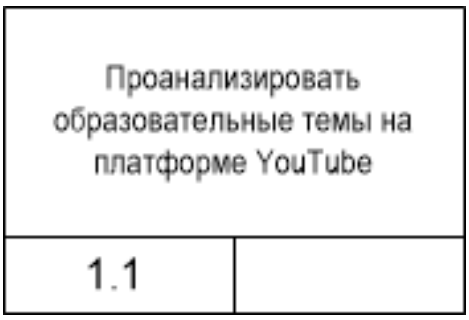

Рисунок 2.1.2.1 – Основная единица работы IDEF3

На следующем шаге необходимо декомпозировать схему 1.1, построенную на прошлом этапе. При декомпозиции необходимо выделить объекты запуска анализов, получения данных от сервисов Google, сохранения полученных данных и формирования итогового отчета, которые будут соответствовать номерам 1.1.2, 1.1.3, 1.1.4 и 1.1.5.

При выполнении анализа система будет последовательно выполнять все виды работ. После этапа получения данных от сервисов Google возможна ситуация отсутствия необходимой для выполнения анализа квоты. В таком случае система прекращает выполнение алгоритма и отправляет пользователю сообщение об отсутствии квоты. Схема декомпозиции основной работы представлена на рисунке 2.1.2.2.

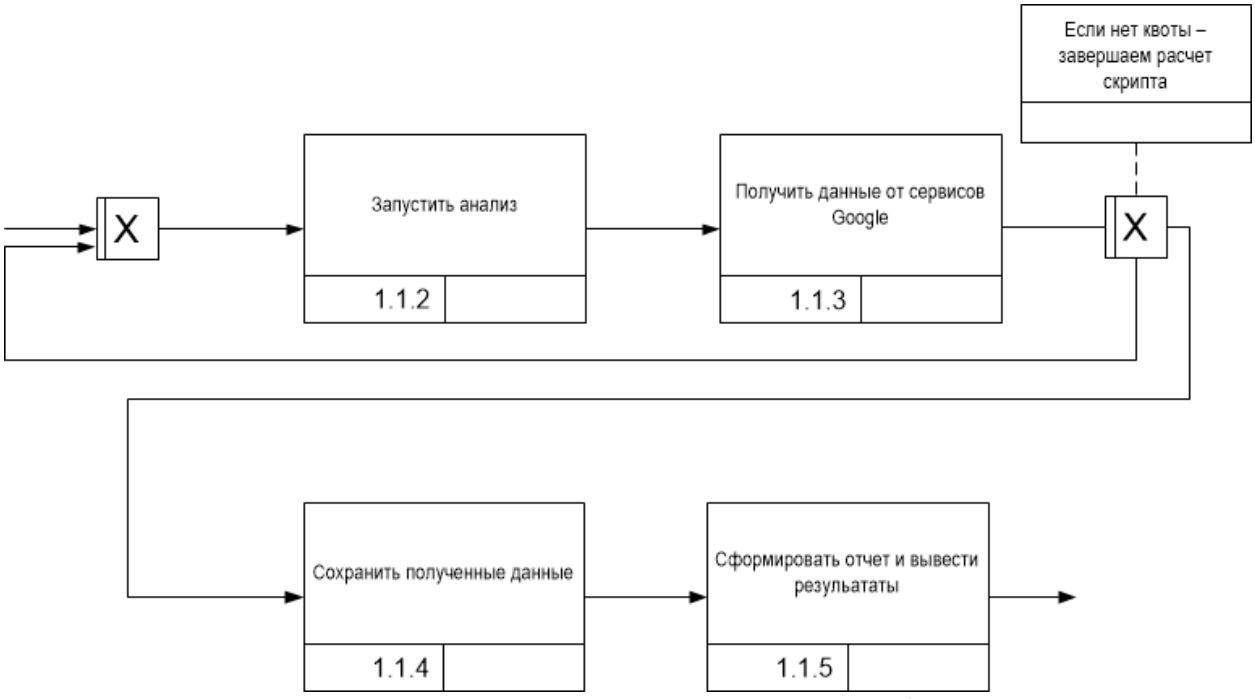

Рисунок 2.1.2.2 – Декомпозиция основной работы IDEF3

На следующем этапе необходимо декомпозировать все представленные выше виды работ. Первым делом декомпозируем работу запуска анализа. Данная схема представляет из себя последовательное выполнение работ по вводу поисковых запросов, генерации электронного

ключа и запуска удаленного скрипта. Схема декомпозиции запуска анализа представлена на рисунке 2.1.2.3.

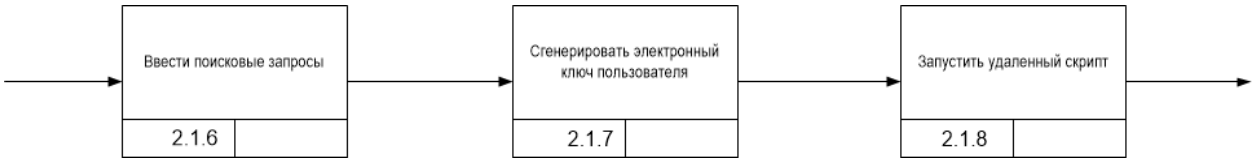

Рисунок 2.1.2.3 – Декомпозиция запуска анализа IDEF3

Далее нужно декомпозировать работу по получению данных от сервисов Google. При получении данных первым делом необходимо проверить квоту API-ключа. Если квоты нет – происходит выход из данного процесса. Если квота есть – производится получение списка всех видеозаписей по каждому запросу и данных о каждой конкретной видеозаписи. Схема декомпозиции работы получения данных от сервисов Google представлена на рисунке 2.1.2.4.

Рисунок 2.1.2.4 – Декомпозиция получения данных от Google IDEF3

На следующем этапе необходимо декомпозировать работу по сохранению полученных данных. Первым делом необходимо создать базу данных для хранения статистики, после этого внести в нее информацию о запросах, о списке видеозаписей и информацию о каждой конкретной видеозаписи. Схема декомпозиции работы сохранения полученных данных представлена на рисунке 2.1.2.5.

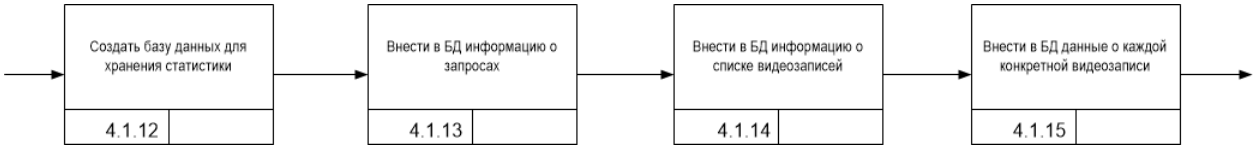

Рисунок 2.1.2.5 – Декомпозиция сохранения полученных данных IDEF3

На следующем шаге необходимо декомпозировать работу по формированию отчета и выводу результатов. В ходе данной работы выполняется генерация графиков, формирование HTML-отчета и вывод HTML-отчета клиенту с соответствующим электронным ключем.

Схема декомпозиции работы формирования HTML-отчета и вывода его клиента представлена на рисунке 2.1.2.6.

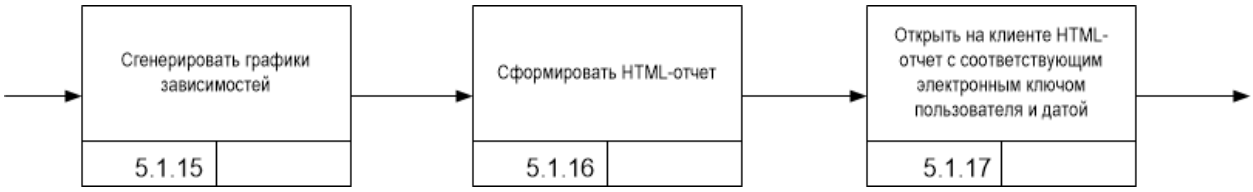

Рисунок 2.1.2.6 – Декомпозиция формирования HTML-отчета IDEF3

На данном этапе были разработаны IDEF3-модели проектируемой информационной системы.

2.1.3 Диаграмма DFD

DFD-модель – это диаграмма потоков данных. Главной целью такого представления является демонстрация того, как каждый процесс преобразует свои входные данные в выходные, а также выявление отношения между данными процессами [3].

Основными компонентами DFD являются:

1) внешние сущности – являются источником или приемником информации, не входят в саму систему;

2) системы и подсистемы – основной компонент данной модели;

3) процессы – представляет собой преобразование входных потоков в выходные, отличается от системы по полю наименования;

4) хранилища данных – представляет собой устройство для хранения информации;

5) потоки данных – показывает, какую информацию передает источник приемнику.

Разрабатываемая система в нотации DFD представлена контекстной диаграммой с основным процессом, представляющим из себя проведение анализа образовательных тем. В качестве внешних сущностей отмечены пользователь, сервисы Google и браузер. Потоками данных на данной диаграмме являются HTML-отчет, поисковые запросы, интернет-протокол и учетная запись Google APIs. Контекстная диаграмма разрабатываемой системы в нотации DFD представлена на рисунке 2.1.3.1.

Рисунок 2.1.3.1 – Контекстная диаграмма DFD

На следующем шаге необходимо провести декомпозицию построенной ранее контекстной диаграммы. В ходе декомпозиции были выделены процессы запуска анализа, получения данных от сервисов Google, сохранения полученных данных и формирования итогового HTML-отчета с его выводов на клиентскую часть. В ходе декомпозиции контекстной диаграммы были выделены следующие хранилища данных: информация о поисковых запросах, параметры скрипта, данные о пользователе, список видео по каждому запросу, реестр статистики по каждой конкретной видеозаписи, база данных со статистикой, архив графиков. К потокам данных можно отнести список поисковых запросов, учетную запись Google APIs, электронный ключ пользователя, список видеозаписей, статистику видео, поисковые запросы, дату и время начала работы скрипта, статистику, графики зависимостей и интернет-протокол.

Декомпозиция контекстной диаграммы разрабатываемой системы в нотации DFD представлена на рисунке 2.1.3.2.

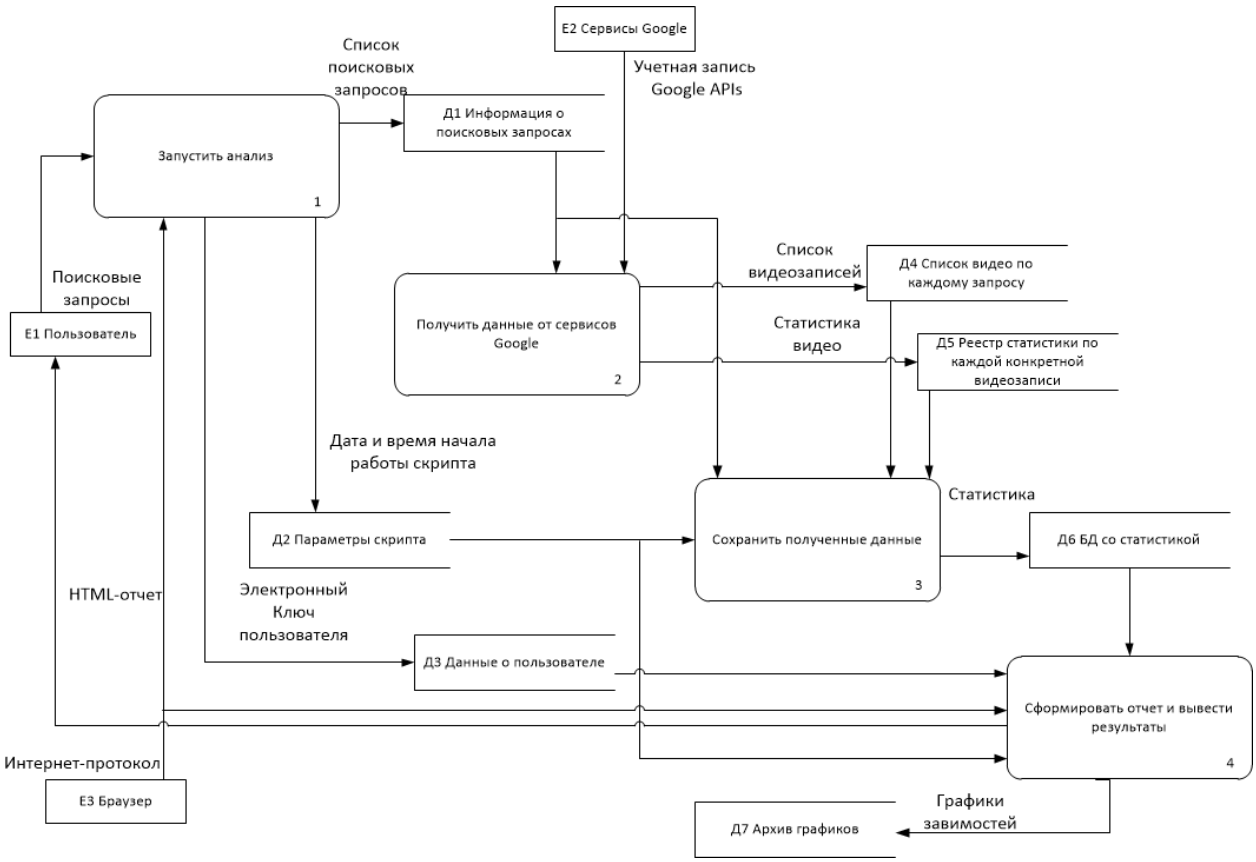

Рисунок 2.1.3.2 – Декомпозиция контекстной диаграммы DFD

Далее необходимо более подробно рассмотреть основные процессы, выделенные на этапе декомпозиции контекстной диаграммы.

Декомпозируем процесс запуска анализа. На данном этапе выделяются процессы ввода поисковых запросов, генерации электронного ключа пользователя и запуск удаленного скрипта.

Декомпозиция процесса запуска анализа в нотации DFD представлена на рисунке 2.1.3.3.

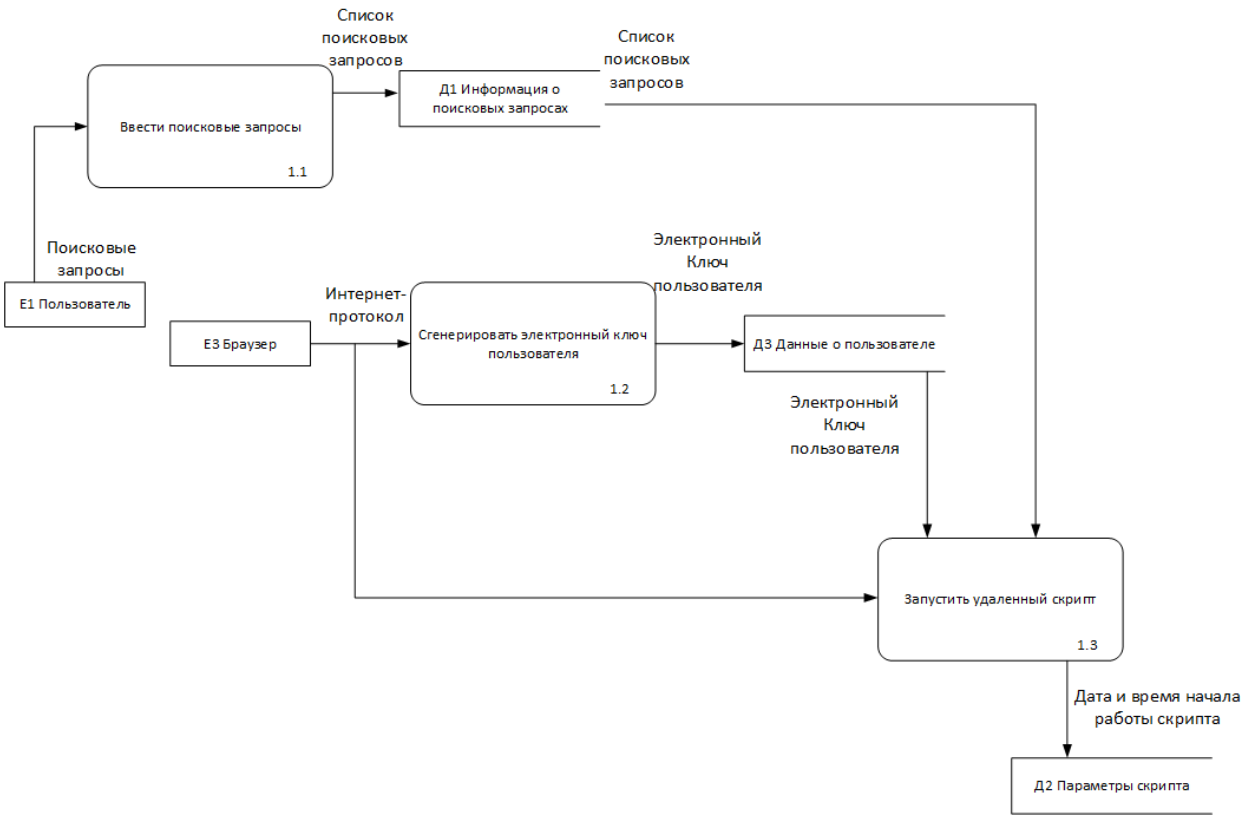

Рисунок 2.1.3.3 – Декомпозиция запуска анализа DFD

На следующем шаге декомпозируем процесс получения данных от сервисов Google. На данном этапе выделяются процессы проверки квоты API-ключа, получения списка всех видеозаписей и получения данных каждой конкретной видеозаписи.

Декомпозиция процесса получения данных от сервисов Google в нотации DFD представлена на рисунке 2.1.3.4.

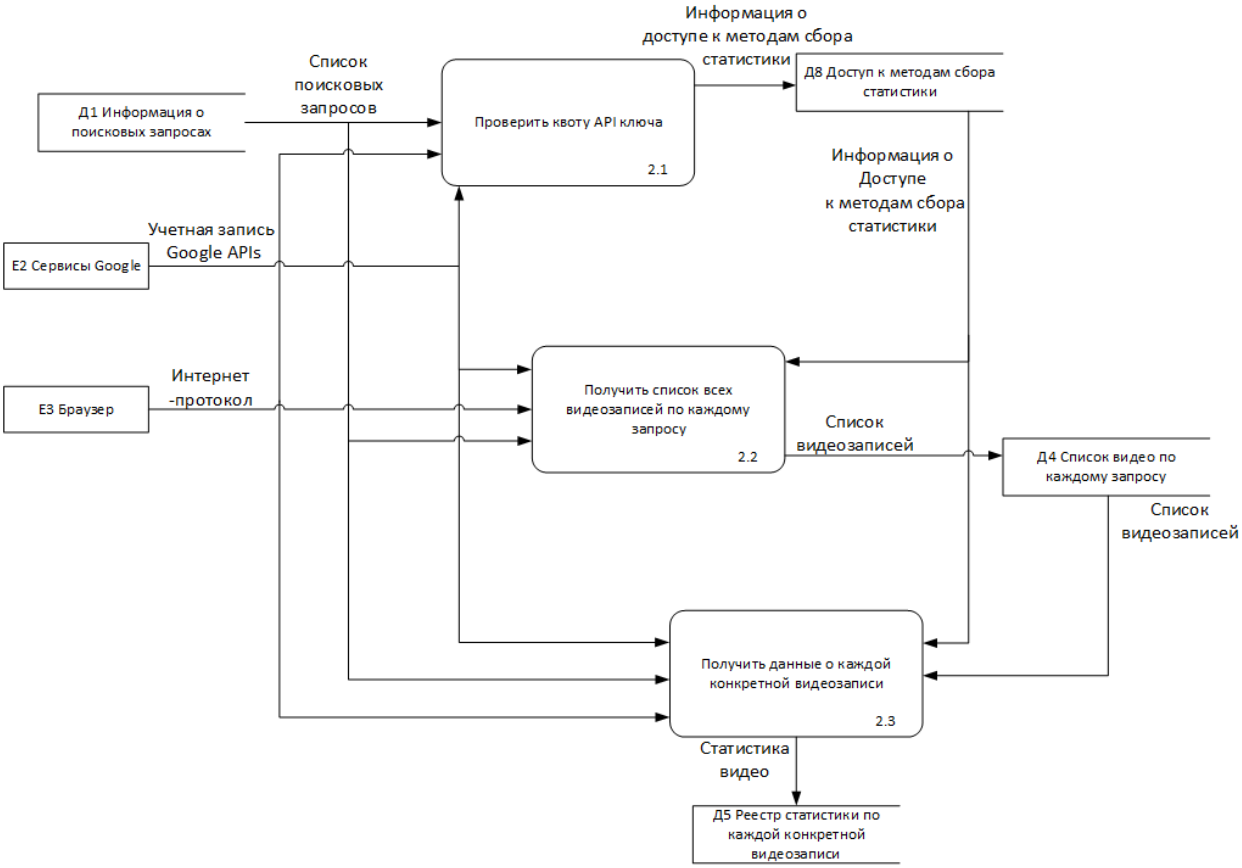

Рисунок 2.1.3.4 – Декомпозиция получения данных от сервисов Google DFD

Далее необходимо декомпозировать процесс сохранения полученных данных. На данном этапе выделяются процессы создания базы данных, внесения в нее информации о запросах, внесения в нее информации о списке видеозаписей и внесение в базу данных информации о каждой видеозаписи.

Декомпозиция процесса сохранения полученных данных в нотации DFD представлена на рисунке 2.1.3.5.

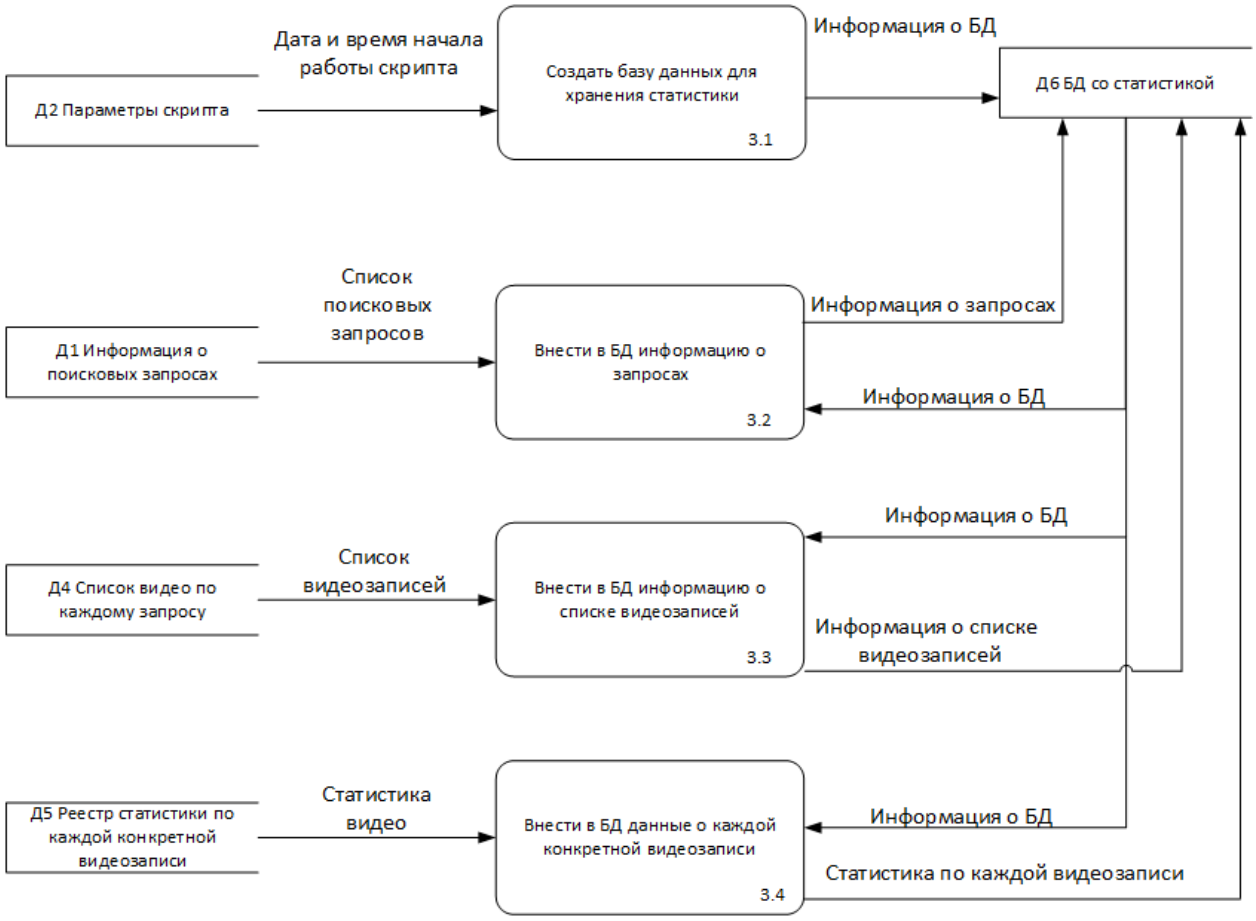

Рисунок 2.1.3.5 – Декомпозиция сохранения полученных данных DFD

После этого необходимо декомпозировать процесс генерации HTML-отчета и вывода его на клиентскую часть. На данном этапе выделяются процессы генерации графиков, формирования HTML-отчета и процесс открытия HTML-отчета с соответствующим электронным ключем пользователя и датой.

Декомпозиция процесса генерации HTML-отчета и вывода его на клиентскую часть в нотации DFD представлена на рисунке 2.1.3.6.

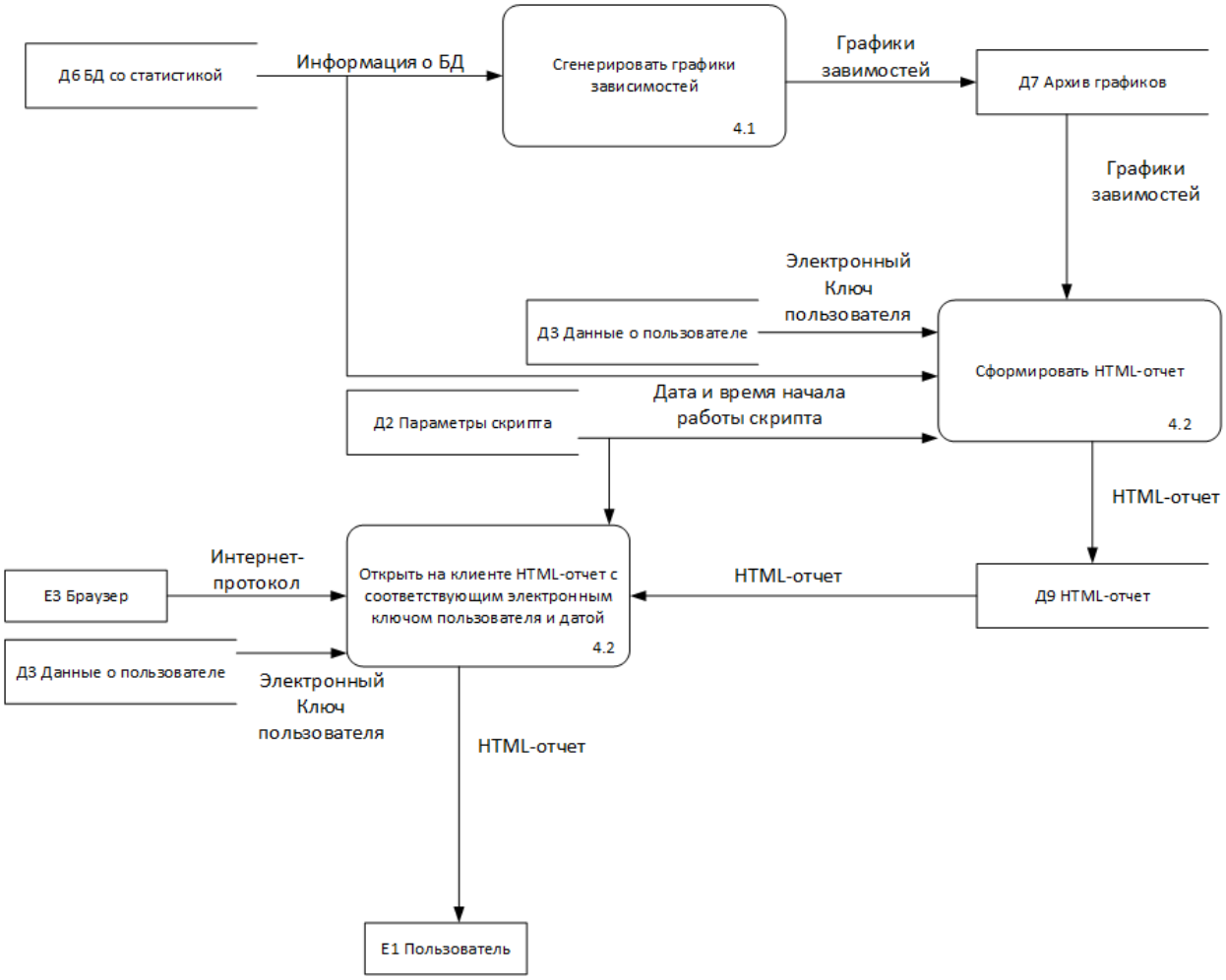

Рисунок 2.1.3.6 – Декомпозиция формирования и вывода HTML-отчета DFD

Таким образом, на данном этапе работы над курсовым проектом были спроектированы DFD-диаграммы, показывающие направления потоков данных в информационной системе.

2.2 Разработка моделей данных

Модели данных служат для проектирования структуры постоянных хранилищ данных, используемых системой. При проектировании информационной системы автоматизации аналитики роликов YouTube в интересах образовательного учреждения были разработаны логическая и физическая модели данных.

2.2.1 Логическая модель

Целью построения логической модели является получение графического представления логической структуры исследуемой предметной области.

Логическая модель предметной области иллюстрирует сущности, а также их взаимоотношения между собой.

Сущности описывают объекты, являющиеся предметом деятельности предметной области, и субъекты, осуществляющие деятельность в рамках предметной области. Свойства объектов и субъектов реального мира описываются с помощью атрибутов.

Взаимоотношения между сущностями иллюстрируются с помощью связей. Правила и ограничения взаимоотношений описываются с помощью свойств связей. Обычно связи определяют либо зависимости между сущностями, либо влияние одной сущности на другую [4].

При проектировании информационной системы автоматизации аналитики роликов YouTube были выделены 3 основные сущности:

- результат;
- запрос;
- видео.

Разработанная логическая модель с указанием связей между сущностями представлена на рисунке 2.2.1.1.

Рисунок 2.2.1.1 – Логическая модель

Сущность «Результат» имеет следующие атрибуты: «Код результата», «Количество видео», «Количество лайков», «Количество дизлайков», «Количество комментариев», «Количество просмотров», «Код запроса».

Сущность «Запрос» характеризуется атрибутами «Код запроса» и «Текст запроса».

Сущность «Видео» имеет следующие атрибуты: «Код видео», «Ссылка», «Embed-тэг», «Количество лайков», «Количество дизлайков», «Количество комментариев», «Количество просмотров», «Дата», «Название», «Код запроса».

2.2.2 Физическая модель

Физическая модель – логическая модель базы данных, выраженная в терминах языка описания данных конкретной СУБД. Главным отличием физической модели от логической заключается в наличии типов у атрибутов сущностей.

В качестве СУБД была выбрана SQLite, так как она лучше всех подходит под требования проекта. Файловая структура данной СУБД состоит из одного файла, поэтому ее легко переносить на другие машины, помимо этого данная СУБД является очень экономичной, в плане ресурсов, архитектурой и имеет высокую скорость выполнения простых операций выборки данных.

Разработчики SQLite3 максимально позаботились о совместимости баз данных SQLite3 с другими СУБД, для этого они ввели аффинированные типы данных для столбцов. Аффинированный тип данных – это рекомендуемый тип данных для столбца. Конечно, это не отменяет того, что в любой столбец можно записать любое значение, но в некоторых ситуациях СУБД будет отдавать приоритет аффинированному типу данных и стараться преобразовать значения к рекомендуемому типу [5].

При создании таблиц мы можем присваивать аффинированный тип данных для столбца, аффинированных типов данных в SQLite3 всего пять:

- blob;
- text;
- real;
- integer;
- numeric.

При разработке физической модели были использованы базовые типы integer, text и blob. Стоит отметить тип blob, суть которого заключается в том, что он не предпринимает никаких попыток принудительно преобразовать полученные данные к какому-то определенному типу и хранит данные в том же виде, в котором они были получены. В нашем случае тип blob использовался для хранения значения даты выпуска видео.

Разработанная физическая модель показана на рисунке 2.2.2.1.

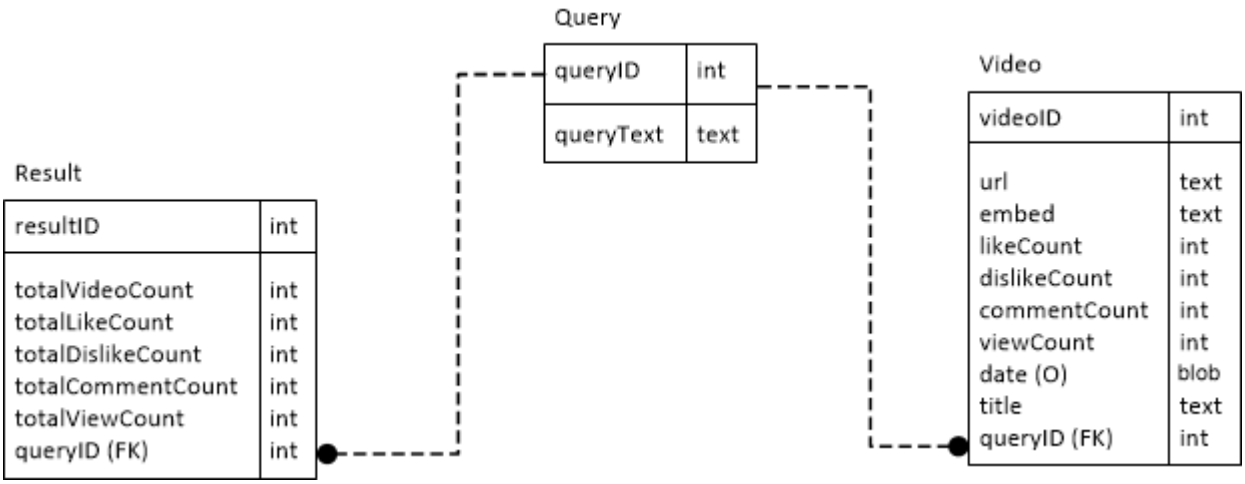

Рисунок 2.2.2.1 – Физическая модель

Описание характеристик столбцов для таблиц «Result», «Query» и «Video» представлены в таблицах 2.2.2.1, 2.2.2.2 и 2.2.2.3.

Таблица 2.2.2.1 – Таблица «Result»

Атрибут	Тип	PK	FK	NOT NULL	UNIQUE	Cascade delete, update
resultID	integer	+	-	+	+	-
totalVideoCount	integer	-	-	+	-	-
totalLikeCount	integer	-	-	+	-	-
totalDislikeCount	integer	-	-	+	-	-
totalCommentCount	integer	-	-	+	-	-
totalViewCount	integer	-	-	+	-	-
queryID	integer	-	+	+	-	+

Таблица 2.2.2.2 – Таблица «Query»

Атрибут	Тип	PK	FK	NOT NULL	UNIQUE	Cascade delete, update
queryID	integer	+	-	+	+	-
queryText	text	-	-	+	-	-

Таблица 2.2.2.3 – Таблица «Video»

Атрибут	Тип	PK	FK	NOT NULL	UNIQUE	Cascade delete, update
videoID	integer	+	-	+	+	-
url	text	-	-	+	-	-
embed	text	-	-	+	-	-
likeCount	integer	-	-	+	-	-
dislikeCount	integer	-	-	+	-	-
commentCount	integer	-	-	+	-	-
viewCount	integer	-	-	+	-	-
date	blob	-	-	-	-	-
title	text	-	-	+	-	-
queryID	integer	-	+	+	-	+

В таблицах использованы следующие обозначения:

- PK (primary key) – представляет собой индекс и применяется для уникальной идентификации записей таблицы. Никакие из двух записей таблицы не могут иметь одинаковых значений первичного ключа.
- FK (foreign key) – внешний ключ, обеспечивают логическую связь между таблицами реляционной базы данных.
- NOT NULL – специальное значение, которое может быть записано в поле таблицы БД. NOT NULL соответствует понятию «не пустое поле». СУБД не разрешает значение NULL для полей, являющихся частью первичного ключа таблицы. В полях внешних ключей NULL допускается.
- UNIQUE – уникальное поле, ограничения UNIQUE можно использовать для обеспечения того, чтобы в указанные столбцы, не входящие в состав первичного ключа, не вводились повторяющиеся значения.
- CHECK – обеспечивает ограничение, которое позволяет установить условие, которому должно удовлетворять значение, вводимое в таблицу, прежде чем оно будет принято.

- Cascade delete, update – каскадное удаление и обновление. Позволяет удалять связанные данные из зависимой таблицы, при удалении данных из основной таблицы.

2.3 Выводы к разделу 2

В рассмотренном выше разделе произведено IDEF0, DFD и IDEF3-моделирование, которое позволяет подробно описать все информационные объекты, содержащиеся в системе, назначение самой ИС и взаимосвязи с ее внутренними и внешними элементами.

Помимо этого, в данном разделе были спроектированы модели данных, показывающие структуру основного хранилища данных, используемого системой.

3 Объектно-ориентированное проектирование

На данном этапе разработки информационной системы произведено UML-проектирование. UML – это унифицированный язык моделирования, применяемый для объектно-ориентированного анализа и проектирования. UML используется для визуализации, конструирования и документирования программных средств [6].

Разрабатываемая информационная система автоматизации аналитики роликов YouTube будет описываться при помощи диаграммы вариантов использования, диаграммы деятельности, диаграммы последовательности, диаграммы классов, диаграммы состояний и диаграммы компонентов.

3.1 Диаграмма вариантов использования (Use Case)

Диаграмма вариантов использования описывает функциональное назначение системы, то есть что система будет делать в процессе своего функционирования.

Use Case является исходной концептуальной моделью системы в процессе ее проектирования и разработки.

Цели построения диаграммы Use Case:

- Определение общих границ и контекста моделируемой предметной области на начальных этапах проектирования.
- Формулирование общих требований к функциональному проектированию системы.
- Разработка исходной концептуальной модели системы для ее последующей реализации.
- Подготовка документации для взаимодействия разработчика системы с ее заказчиком и пользователями.

В ходе проектирования система представляется в виде множества актеров, взаимодействующих с системой с помощью прецедентов.

Таким образом, основными компонентами диаграммы вариантов использования являются:

- Актеры – взаимодействуют с системой и используют ее функциональные возможности для достижения определенных целей и решения частных задач. Представляют собой внешнюю по отношению к моделируемой системе сущность. Может рассматриваться как некая роль относительно конкретного варианта использования.

- Прецеденты – определяют последовательность действий, которую должна выполнять система при взаимодействии ее с соответствующим актером.

- Отношения – показывают тип взаимодействия актеров и прецедентов. Один актер может взаимодействовать с несколькими вариантами использования и наоборот.

Существует 4 вида отношений между актерами и прецедентами:

- Ассоциативное отношение – устанавливает какую конкретную роль актер играет при взаимодействии с вариантом использования.

- Отношение расширения – определяет взаимосвязь базового варианта использования с некоторым другим вариантом использования, функциональное поведение которого задействуется базовым не всегда, а при выполнении некоторых дополнительных условий.

- Отношение обобщения – служит для указания того факта, что некоторый вариант использования может быть обобщен до другого варианта использования.

- Отношение включения – указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования.

Разработанная диаграмма вариантов использования представлена на рисунке 3.1.1.

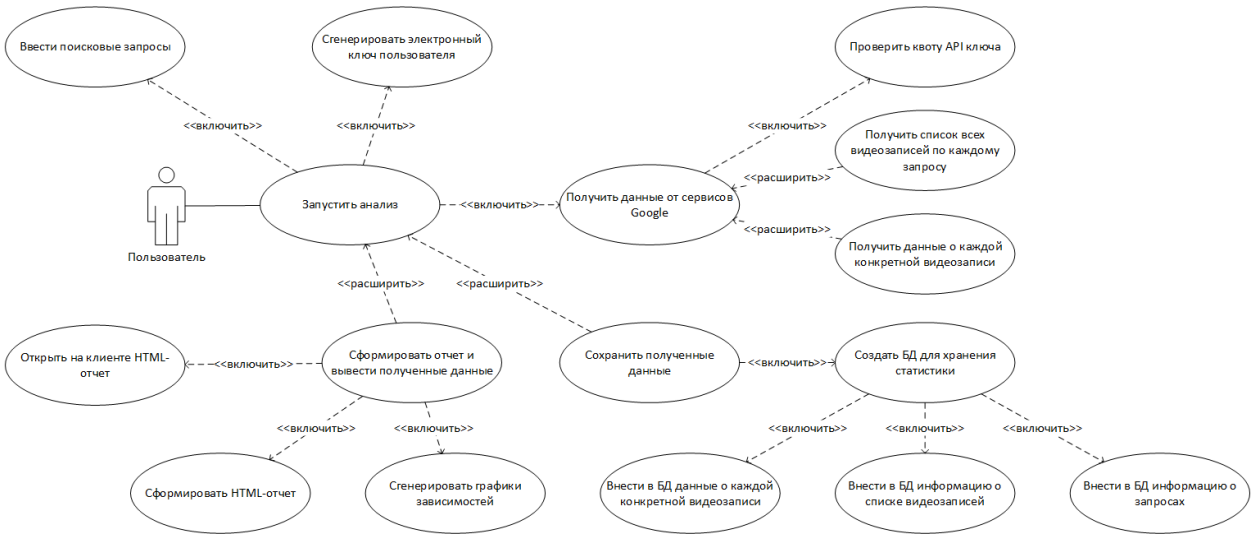

Рисунок 3.1.1 – Диаграмма вариантов использования

В ходе проектирования информационной системы автоматизации анализа роликов на платформе YouTube был выделен один актер – «Пользователь» и один основной прецедент, взаимодействующий с актером – «Запустить анализ».

Прецедент запуска анализа включает в себя варианты использования «Ввести поисковые запросы», «Сгенерировать электронный ключ пользователя» и «Получить данные от сервисов Google». Это обусловлено

тем, что данные варианты использования будут обязательно выполнятся при запуске анализа.

Помимо этого, прецедент «Запуск анализа» имеет расширения «Сохранить полученные данные» и «Сформировать отчет и вывести полученные данные», которые выполняются только при условии наличия квоты API-ключей.

Диаграмма вариантов использования системы содержит все рассмотренные выше прецеденты, а также включающие и расширяющие их варианты использования.

3.2 Диаграмма деятельности (Activity)

Диаграмма деятельности отражает динамику системы и представляет собой схемы потоков управления в системе от действия к действию, а также параллельные действия и альтернативные потоки.

В контексте языка UML деятельность представляет собой некоторую совокупность отдельных вычислений, выполняемых автоматом.

К основным компонентам диаграммы деятельности относятся:

- Действие – выполнение какой-либо функции.
- Переход – отражает соединение действий.
- Элемент выбора – некоторое условие, предназначенное для разветвления диаграммы.
- Линия синхронизации – линия, служащая для слияния и разветвления диаграммы на параллельные процессы.

Диаграмма деятельности разрабатываемой информационной системы представлена на рисунке 3.2.1.

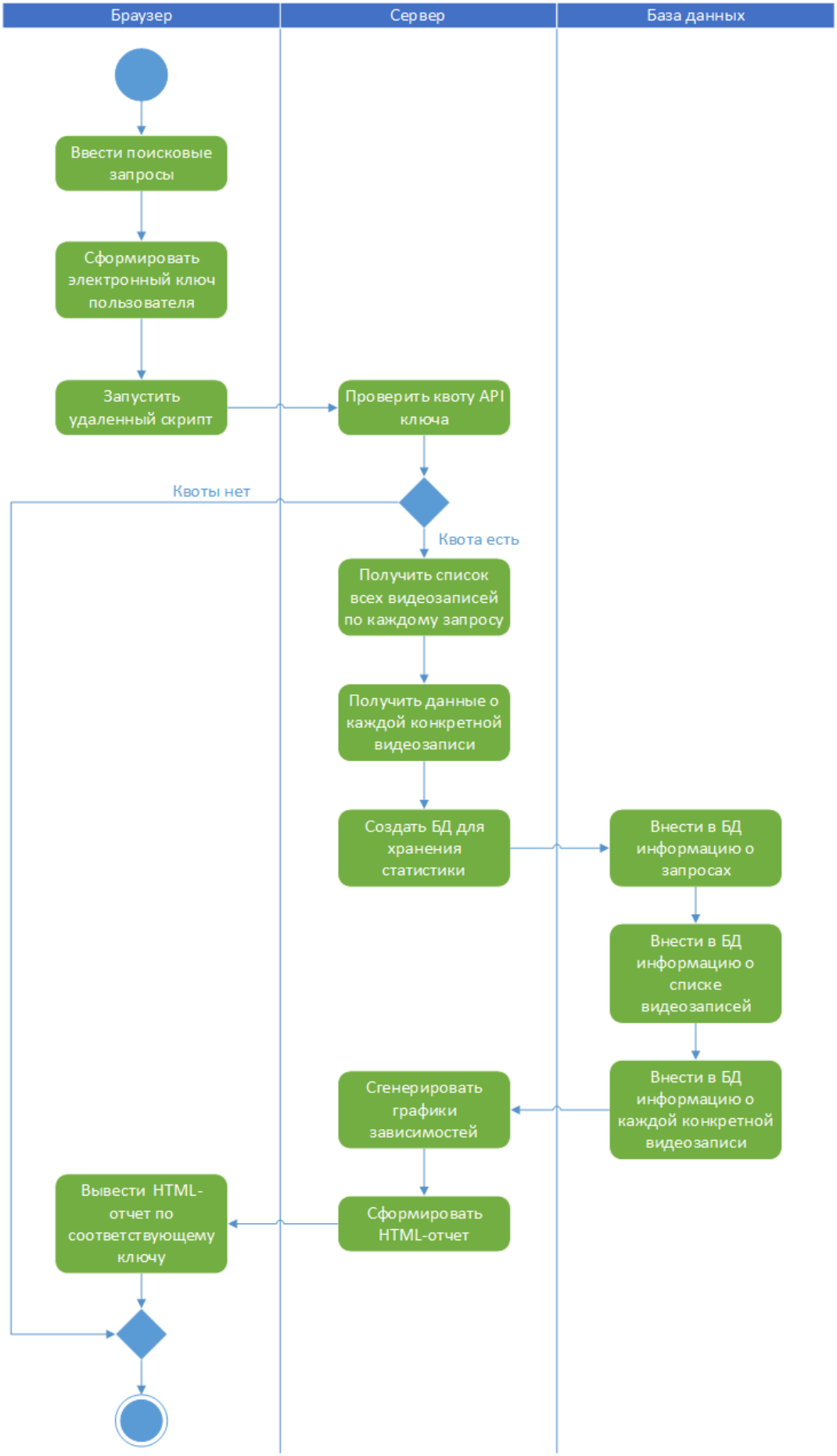

Рисунок 3.2.1 – Диаграмма деятельности

Стоить отметить, что при построении диаграммы деятельности системы автоматизации аналитики роликов YouTube использовались дорожки, с помощью которых обозначалось на каком из модулей системы (браузер, сервер, база данных) выполнялось конкретное действие.

3.3 Диаграмма последовательности (Sequence)

Диаграммы последовательности действий отображают взаимодействие объектов, упорядоченное во времени.

К основным компонентам диаграммы последовательности можно отнести:

- Объекты – экземпляры класса.
- Линия жизни – вертикальная линия, которая показывает создание и уничтожение объекта, а также на ней находится фокус управления.
- Сообщения – законченный фрагмент информации, который отправляется одним объектом другому.

Стоит учесть, что диаграммы последовательности строятся на основе диаграмм деятельности, а именно каждая диаграмма последовательности какого-либо процесса должна соответствовать одной диаграмме активности этого же процесса.

В ходе проектирования информационной системы автоматизации аналитики роликов YouTube была построена диаграмма последовательности системы, которая представлена на рисунке 3.3.1.

В данной диаграмме выделены 5 объектов: пользователь, web-браузер, web-сервер, cgi-скрипт и база данных. Эта диаграмма полностью описывает процесс работы системы в соответствии с построенными ранее диаграммами Use Case и Activity и отображает все взаимодействия и жизненные циклы выделенных объектов.

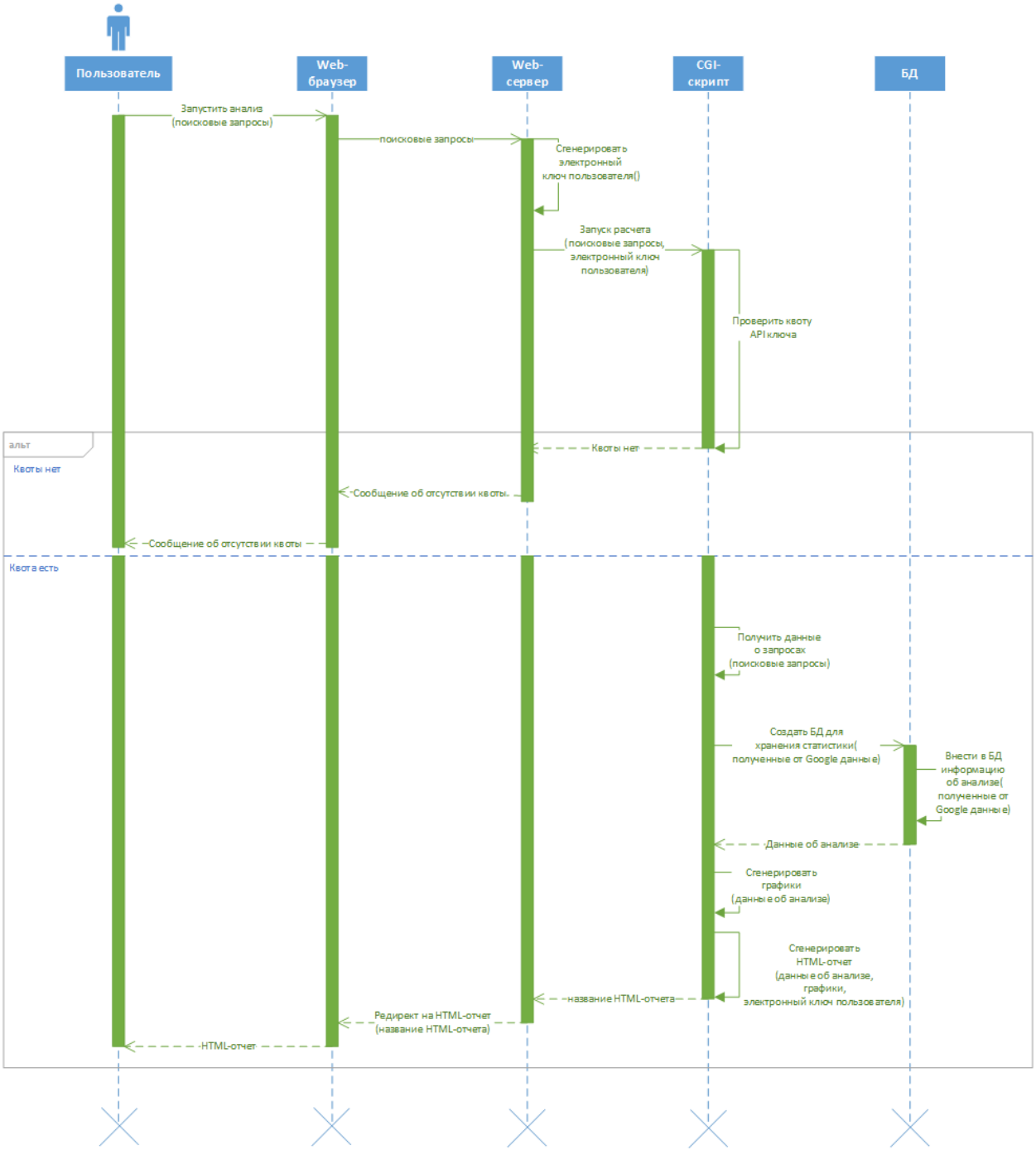

Рисунок 3.3.1 – Диаграмма последовательности системы

Таким образом, в данном подразделе курсового проекта была построена диаграмма последовательности информационной системы автоматизации аналитики роликов на платформе YouTube.

3.4 Диаграмма классов (Class)

Диаграмма классов является центральным звеном объектно-ориентированного подхода и отражает основную информацию об объектах системы и связях между ними.

К основным видам связей относятся:

- Отношения ассоциации – свидетельствует о наличии произвольного отношения между классами.
- Отношения обобщения – является отношением классификации между более общим элементом (родителем или предком) и более частным (дочерним или потомком).
- Отношения агрегации – показывает включение некоторой сущности в качестве составной части другой сущности.
- Отношения композиции – является частным случаем агрегации, при условии, что части не могут выступать в отрыве от целого, то есть с уничтожением целого уничтожаются составные части.
- Отношения зависимости – используется в такой ситуации, когда некоторое изменение одного элемента модели может потребовать изменения другого элемента.

На основе диаграммы последовательности и рассмотренных выше видов связей была спроектирована диаграмма классов разрабатываемой информационной системы, которая представлена на рисунке 3.4.1.

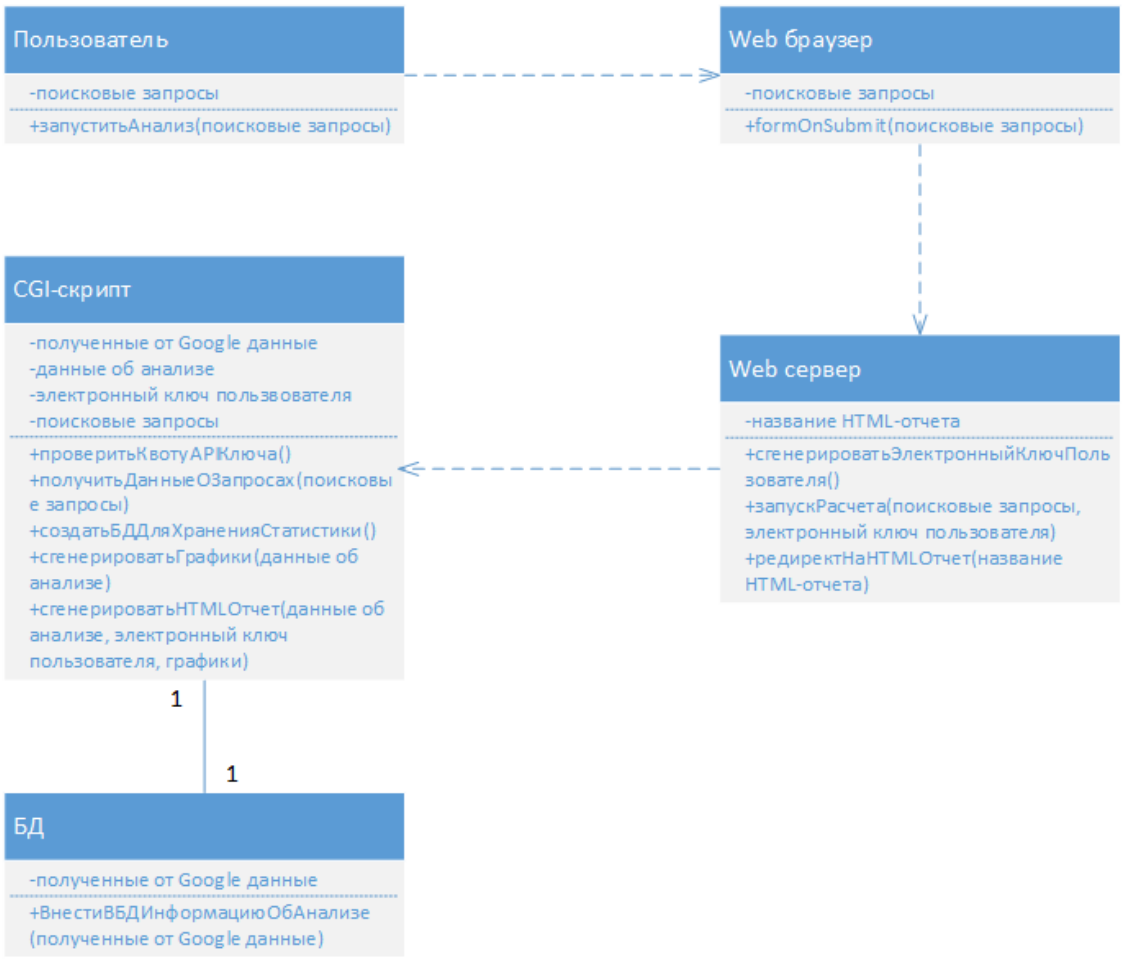

Рисунок 3.4.1 – Диаграмма классов

В ходе проектирования диаграммы классов разрабатываемой информационной системы были использованы отношения агрегации и зависимости.

3.5 Диаграмма состояния (State)

Диаграмма состояний описывает все возможные состояния, в которых может находиться объект, а также процесс смены состояний в результате наступления некоторого события.

К основным элементам диаграммы состояний относятся:

- Состояния – представляет собой местонахождение управления диаграммы состояний
- Переходы – отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния другим.

На основе построенных ранее диаграм и рассмотренной выше теории была построена диаграмма состояний разрабатываемой системы представлена на рисунке 3.5.1.

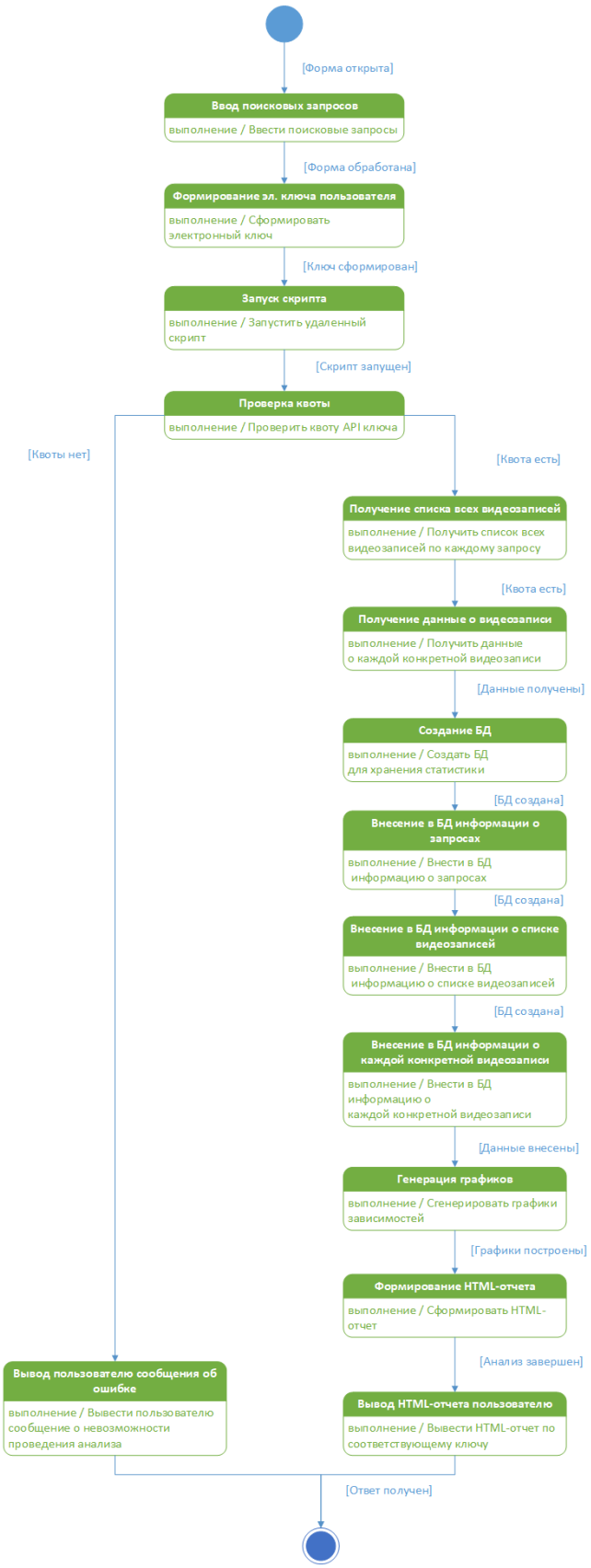

Рисунок 3.5.1 – Диаграмма состояний

Таким образом, в данном подразделе курсового проекта была построена диаграмма состояний разрабатываемой информационной системы.

3.6 Диаграмма компонентов (Component)

Диаграмма компонентов — статическая структурная диаграмма, показывает разбиение системы на структурные компоненты и связи между ними. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы и пакеты. Такой вид диаграммы позволяет переходить от логического представления системы к ее реализации в виде программного кода [7].

Для представления физических сущностей в языке UML применяется термин – компонент. Компонент реализует некоторый набор интерфейсов и служит для общего обозначения элементов физического представления модели.

Компонент может иметь также свои собственные свойства, такие как атрибуты и операции.

Диаграмма компонентов разрабатываемой системы представлена на рисунке 3.6.1.

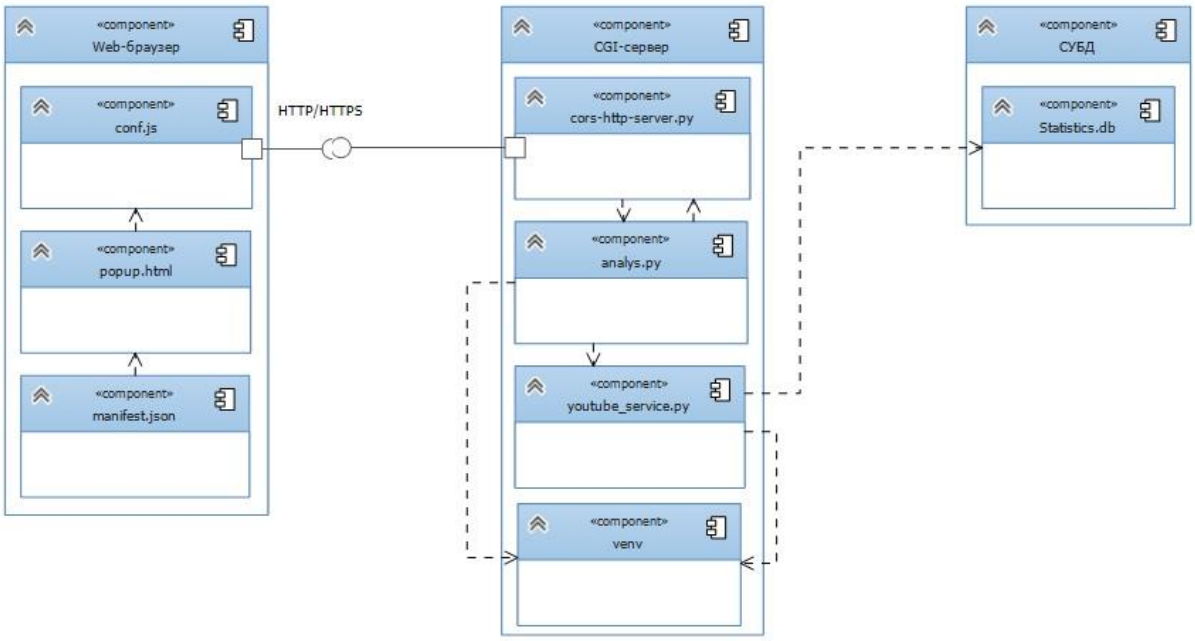

Рисунок 3.6.1 – Диаграмма компонентов

На данном этапе работы над курсовым проектом была спроектирована диаграмма компонентов, показывающая основные структурные компоненты информационной системы и связи между ними.

3.7 Разработка программного обеспечения

В ходе работы над курсовым проектом были реализованы некоторые функции проектируемой информационной системы. На данном этапе была

спроектирована форма взаимодействия пользователя и системы, данная форма представлена на рисунке 3.7.1.

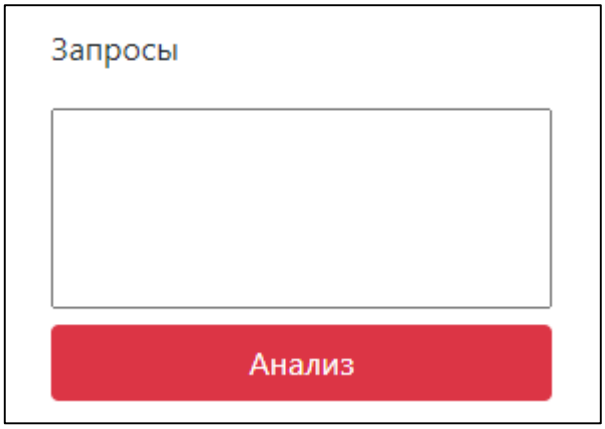

Рисунок 3.7.1 – Экранная форма взаимодействия с пользователем

Помимо разработанной формы взаимодействия с пользователем, был частично реализован алгоритм аналитики видеороликов на платформе YouTube, а также реализована функция составления итогового HTML-отчета. Экранные формы итогового HTML-отчета представлены на рисунках 3.7.2 – 3.7.5.

Отчет об анализе популярности тематических роликов на YouTube

на четверг 17 сентября 2020г.

Количество видео по категориям
- **подготовка к егэ по обществознанию:** 119017 в.
- **deep learning:** 1000000 в.
- **blockchain:** 1000000 в.

Количество лайков по категориям
- **подготовка к егэ по обществознанию:** 10466 л.
- **deep learning:** 22569 л.
- **blockchain:** 36995 л.

Количество дизлайков по категориям
- **подготовка к егэ по обществознанию:** 214 д.
- **deep learning:** 397 д.
- **blockchain:** 1027 д.

Количество комментариев по категориям
- **подготовка к егэ по обществознанию:** 735 к.
- **deep learning:** 1821 к.
- **blockchain:** 6235 к.

Количество просмотров по категориям
- **подготовка к егэ по обществознанию:** 183547 п.
- **deep learning:** 477477 п.
- **blockchain:** 546456 п.

Среднее количество лайк/просмотр
- **подготовка к егэ по обществознанию:** 0.05702 л/п
- **deep learning:** 0.04727 л/п
- **blockchain:** 0.0677 л/п

Среднее количество дизлайк/просмотр
- **подготовка к егэ по обществознанию:** 0.00117 д/п
- **deep learning:** 0.00083 д/п
- **blockchain:** 0.00188 д/п

Отношение лайков к дизлайкам
- **подготовка к егэ по обществознанию:** 48.9 л/д
- **deep learning:** 56.8 л/д
- **blockchain:** 36.0 л/д

Количество видео за последние полгода
- **подготовка к егэ по обществознанию:** 97 в.
- **deep learning:** 100 в.
- **blockchain:** 100 в.

Рисунок 3.7.2 – Основные характеристики запросов

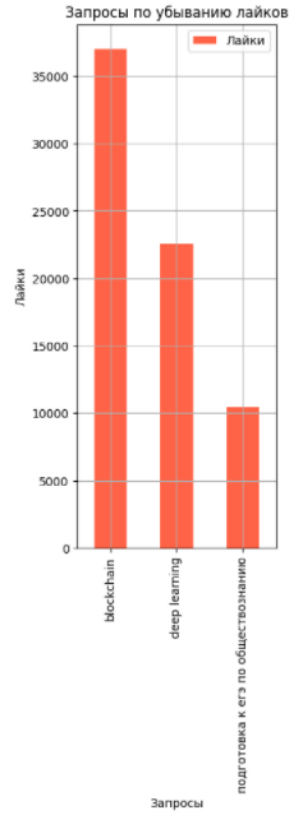

Рисунок 3.7.3 – График запросов по убыванию лайков

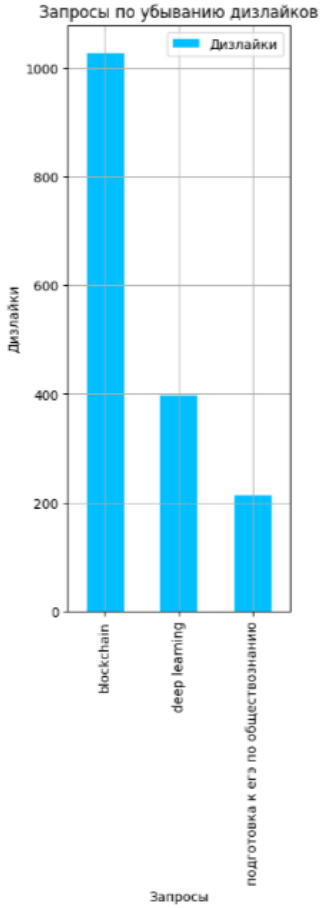

Рисунок 3.7.4 – График запросов по убыванию дизлайков

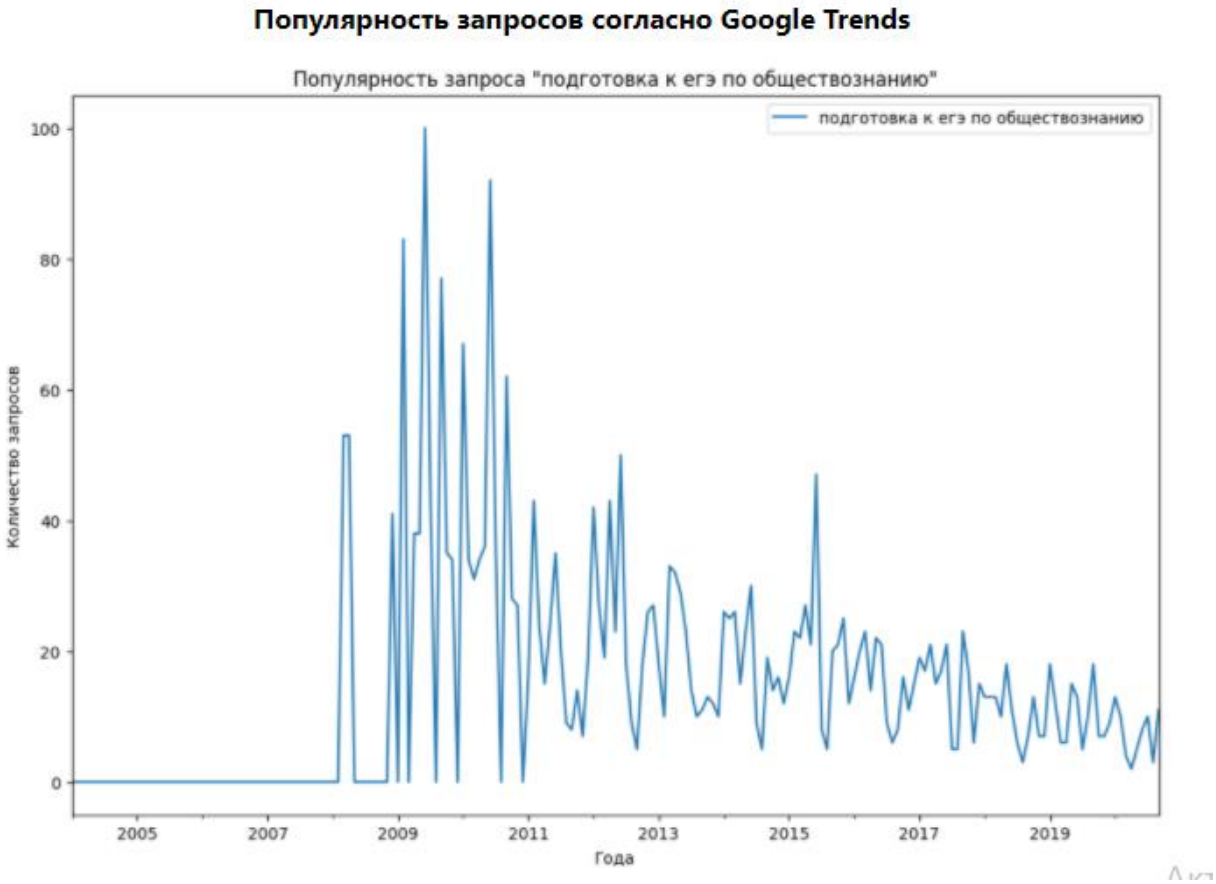

Рисунок 3.7.5 – График популярности запроса «подготовка к егэ по обществознанию»

Код разработанного программного обеспечения представлен в приложении А.

Помимо этого, была написана статья, описывающая разрабатываемый инструмент аналитики видеороликов YouTube. Данная статья опубликована в научном журнале «Студенческий вестник» №2(100), часть 2, 2020 год. Страницы журнала, содержащие текст статьи, представлены на рисунках Б.1 – Б.4 в приложении Б.

3.8 Выводы к разделу 3

В данном разделе было произведено UML-моделирование разрабатываемой информационной системы автоматизации аналитики роликов на платформе YouTube, а именно были построены диаграмма вариантов использования, диаграмма деятельности, диаграмма последовательности, диаграмма классов, диаграмма состояний и диаграмма компонентов.

Помимо этого, в данном разделе была проведена разработка некоторых функций информационной системы, а также опубликована статья в научном журнале «Студенческий вестник», которая описывает разрабатываемый программный продукт.

Заключение

В ходе работы над курсовым проектом были проанализированы предметная область и определены требования к информационной системе, что помогло при дальнейшем ее проектировании.

Помимо этого, были построенные структурные диаграммы, которые включают в себя диаграммы IDEF0, DFD и IDEF3. Данные структурные диаграммы необходимо для полного описания всех объектов системы и их взаимосвязей. Также была построена IDEF1x-диаграмма, показывающая структуру хранилища данных разрабатываемого программного продукта.

Основной частью работы над курсовым проектом является UML-проектирование, в ходе которого были построены диаграмма вариантов использования, диаграмма деятельности, диаграмма последовательности, диаграмма классов, диаграмма состояний и диаграмма компонентов.

Также, были реализованы некоторые функции программного обеспечения, и опубликована статья в журнале «Студенческий вестник».

При дальнейшей работе с проектом планируется доработать пользовательский интерфейс, в том числе окно ввода запросов и итоговый отчет. Помимо этого, планируется реализовать несколько серверных функций, которые еще не разработаны.

Приложение А
(обязательное)
Листинг программного кода

manifest.json:

```
{
  "manifest_version": 2,

  "name":"YouTube Study Analytics",
  "description": "Данное расширение запускает Python-скрипт по анализу роликов YouTube в интересах образовательного учреждения",
  "version": "1.0",
  "content_security_policy"        :        "script-src        'self'        popup.html https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js  'unsafe-eval'  'unsafe-inline'; object-src 'self'",

  "browser_action": {
    "default_popup": "popup.html"
},
"icons": {
      "16": "images/icon.png",
      "32": "images/icon.png",
      "48": "images/icon.png",
      "128": "images/icon.png"
},
  "permissions": [
    "activeTab",
    "https://ajax.googleapis.com/",
    "https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js",
    "https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js",
    "http://localhost:8000/cgi-bin/100videos.py",
    "http://localhost:8000/cgi-bin/script.py"
]
}
```

popup.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>YouTube Study Analytics</title>
<meta charset="UTF-8">
<meta http-equiv="Content-Type" content="text/html" charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link                                                                    rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="./httpPost.js" type="text/javascript" charset="utf-8"></script>
<script          src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
crossorigin="anonymous"></script>
<style>
body {
 margin: 0 !important;
 padding: 0 !important;
 width: 800;
}
```

```
</style>
</head>
<body>
<form id="form" method="POST" style="width: 300px; height: 200px">
    <p style="margin-top: 10px; margin-left: 25px;">Запросы</p>
    <input id="in1" style="width: 250px; position: fixed; margin-top: 2px; margin-left:
25px; height: 100px">
    <p id="in2"></p>
    <button id="btn" class="btn btn-danger" style="width: 250px; position: fixed;
margin-top: 110px; margin-left: 25px;">Анализ</button>
</form>
</body>
</html>
```

httpPost.js:

```
$(document).ready(function(){
                $('#form').submit(function(event){
                    $.post("http://localhost:8000/cgi-bin/100videos.py",
                    {
                     dataType: 'text',
                     data:$("#in1").val()},
                     onResponse);
                    return false;
                })
                function onResponse(data){
                     $("#in2").text(data);
                    console.log(data);
                    window.open("http://localhost:8000/"+data+"/"+data+".html",
'_blank');
                }
            })
```

100videos.py:

```
#import apiclient as api
from googleapiclient import discovery
import sys
import youtube_service as ys
import os
from ctypes import windll
from shutil import rmtree

DEVELOPER_KEYS=["AIzaSyDJR3-A7UnPK6ZVPmYPvUfc35iEjb9TqFk",
                "AIzaSyBCNojrr4-HL23k0sGMMg7OhlDFOZvyTX4",
                "AIzaSyDPs5drcMfvcRrqqkUrhPgnI647438WsdY",
                "AIzaSyA506GzoveUGAvXUib6Y8KTXAJxa4XMdLA",
                "AIzaSyD-o3TMJ0nD--tmSmKp3t1-r88mI6Bc72c",
                "AIzaSyCzKUiwJ7WbbsQqJD7H_QAWNaUB0zzPcoA",
                "AIzaSyBD8tXjVqTDIvcG98zkvs44HS3xWIf_0io",
                "AIzaSyCRHip38suqZie3s6VarjMzTdmSK6E6pDQ"]
YOUTUBE_API_SERVICE_NAME = "youtube"
YOUTUBE_API_VERSION = "v3"
#иконка для модального окна квоты
ICON_EXCLAIM=0x30
#получаем имя БД в формате ДЕНЬ_НЕДЕЛИ_ДД.ММ.ГГГГ_ЧЧ-ММ-СС, с которым потом постоянно
работаем
dbname=ys.getDbName()
#создаем рабочую директорию, в которой создаем еще одну директорию для будущих картинок
root=os.path.abspath(os.curdir)+'\\'+dbname.replace('.db','')+'\\'
path=os.path.abspath(os.curdir)+'\\'+dbname.replace('.db','')+'\\images'
path1='../'+dbname.replace('.db','')+'/images'
```

```
os.mkdir(os.path.abspath(os.curdir)+'\\'+dbname.replace('.db',''))
os.mkdir(path)
print(path)

#создаем новое подключение к БД и получаем курсор для работы с ней
conncurs=ys.createDb(dbname,root)
conn=conncurs[0]
cursor=conncurs[1]

#заполняем БД
ys.fullfillDb(cursor,conn)

#начинаем вывод импровизированного прогрессбара
sys.stdout.write("Сбор и анализ данных [ %d"%0+"% ] ")
sys.stdout.flush()


def msgBox():
    windll.user32.MessageBoxW(0, 'Сбор и анализ данных недоступен, поскольку ежедневная
квота YouTube Data API исчерпана. \n'
                                        'Обнуление квоты произойдёт в 11:00 МСК.',
                                        'Квота исчерпана', 0 | ICON_EXCLAIM)

def youtube_study_analytics():
    totalVideos={}
    totalLikes={}
    totalDislikes={}
    totalComments={}
    totalViews={}

    #построчно читаем запросы из файла
    #здесь кстати можно будет замутить выбор файла пользователем путем ввода его имени

    #sys.argv[1]
    queries=ys.getQueriesFromFile('test.txt')
    #общий главный цикл
    for i in range(len(queries)):
        query=queries[i]
        #вносим изменения в таблицу, коммитим, вопросы это защита от инъекций
        cursor.execute("INSERT INTO query VALUES (?,?)",(i+1,query))
        conn.commit()

        checkQuota=False

        #пробегаемся по АПИ ключам в поисках свободной квоты и выполняем поиск видео по
i-ому запросу
        for h in range(len(DEVELOPER_KEYS)):
                try:
                    youtube          =          discovery.build(YOUTUBE_API_SERVICE_NAME,
YOUTUBE_API_VERSION, developerKey = DEVELOPER_KEYS[h])
                    results  =  youtube.search().list(q  =  query,  part  =  "id,  snippet",
maxResults = 50, order="date").execute()
                    checkQuota=True
                    # print(results)
                    break;
                except:
                    continue

        if checkQuota==False:
            msgBox()
            conn.close()
            rmtree(root,ignore_errors=True)
            return
```

```
            else:
                checkQuota=False

            #выцепляем общее кол-во видео по даннной тематике
            totalVideos[query] = results['pageInfo']['totalResults']
            #получаем токен следующей страницы поиска, если она существует
            if 'nextPageToken' in results:
                nextPageToken=results['nextPageToken']
            else:
                nextPageToken='null'
            #первичная инициализация четырех статистических переменных
            likes,dislikes,comments,views=[0 for y in range(4)]

            #цикл для сбора всех идентификаторов видео
            #и отсеивание лишнего, поскольку в поиске кроме видео есть плейлисты и каналы
            #for k in range(11):
            for k in range(2):
                searchResults=results.get("items", [])
                videoIds=[]
                for result in searchResults:
                    if result['id']['kind']=="youtube#video":
                        videoIds.append(result["id"]["videoId"])
                videoIds=','.join(videoIds)

                #пробегаемся по АПИ ключам в поисках свободной квоты
                #и выполняем сбор данных по всем собранным выше видео
                for j in range(len(DEVELOPER_KEYS)):
                    try:
                        youtube          =          discovery.build(YOUTUBE_API_SERVICE_NAME,
YOUTUBE_API_VERSION, developerKey = DEVELOPER_KEYS[j])
                        results=youtube.videos().list(part                                    =
"snippet,statistics,player",id=videoIds).execute()
                        checkQuota=True
                        break
                    except:
                        continue

                if checkQuota == False:
                    msgBox()
                    conn.close()
                    rmtree(root, ignore_errors=True)
                    return
                else:
                    checkQuota = False

                #собираем все необходимые данные по каждому видео данной тематики
                #лайки, дизлайки, комменты, просмотры, название, дату, ссылку, ссылку для
вставки на сайт
                for videoResult in results.get("items", []):
                    l,d,c,v=[0 for f in range(4)]
                    t=videoResult['snippet']['title']
                    date=videoResult['snippet']['publishedAt']
                    url='https://www.youtube.com/watch?v='+str(videoResult['id'])
                    if 'player' in videoResult:
                        embed=videoResult['player']['embedHtml'].replace('//','https://')
                    if 'statistics' in videoResult:
                        if 'likeCount' in videoResult['statistics']:
                            l=int(videoResult['statistics']['likeCount'])
                            likes=likes+l
                        if 'dislikeCount' in videoResult['statistics']:
                            d=int(videoResult['statistics']['dislikeCount'])
                            dislikes=dislikes+d
                        if 'commentCount' in videoResult['statistics']:
```

```
                        c=int(videoResult['statistics']['commentCount'])
                        comments=comments+c
                    v=int(videoResult['statistics']['viewCount'])
                    views=views+v
                #вносим изменения в таблицу, коммитим
                cursor.execute("INSERT INTO video VALUES
(?,?,?,?,?,?,?,?,?)",(url,embed,t,l,d,c,v,date,i+1))
                conn.commit()

            #читаем следующую страницу поиска, если она существует
            if nextPageToken!='null':
                for j in range(len(DEVELOPER_KEYS)):
                    try:
                        youtube = discovery.build(YOUTUBE_API_SERVICE_NAME,
YOUTUBE_API_VERSION, developerKey = DEVELOPER_KEYS[j])
                        results=youtube.search().list(q = query, part = "id, snippet",
pageToken=nextPageToken, maxResults = 50, order="date").execute()
                        checkQuota=True
                        break
                    except:
                        continue

                if checkQuota == False:
                    msgBox()
                    conn.close()
                    rmtree(root, ignore_errors=True)
                    return
                else:
                    checkQuota = False
            '''if 'nextPageToken' in results:
                nextPageToken=results['nextPageToken']
            else:
                break'''

        #получаем общее кол-во статистических данных по категории
        totalLikes[query]=likes
        totalDislikes[query]=dislikes
        totalComments[query]=comments
        totalViews[query]=views

        #вносим изменения в таблицу, коммитим
        cursor.execute("INSERT INTO result VALUES (?,?,?,?,?,?,?)",
                       (i+1,totalVideos[query],totalLikes[query],totalDislikes[query],
                       totalComments[query],totalViews[query],i+1))
        conn.commit()

        #в конце каждой итерации меняем значение на прогрессбаре
        sys.stdout.write(("\rСбор и анализ данных [ %d"%((i+1)*100/(len(queries)))+"% ]
")+('='*(int((i+1)*10/len(queries)))))
        sys.stdout.flush()
    maxLike=ys.blabla(dbname,root,3)
    maxLikes=maxLike[0]
    maxLikeEmbeds=maxLike[1]
    maxDislike=ys.blabla1(dbname,root,3)
    maxDislikes=maxDislike[0]
    maxDislikeEmbeds=maxDislike[1]
    maxComment=ys.blabla2(dbname,root,3)
    maxComments=maxComment[0]
    maxCommentsEmbeds=maxComment[1]
    maxView=ys.blabla3(dbname,root,3)
    maxViews=maxView[0]
    maxViewsEmbeds=maxView[1]
    queriesEmbed=maxView[2]
```

```
    images=[]
    for i in range(8):
        images.append(path1+'/'+str(i+1)+'.png')

    images1=[]
    for i in range(8,len(queries)+8):
        images1.append(path1+'/'+str(i+1)+'.png')
    images2=[]
    for i in range(len(queries)+8, len(queries)*2+8):
        images2.append(path1+'/'+str(i+1)+'.png')

    print (ys.blabla(dbname,root,3))
    #генерируем кучу графиков и сохраняем их в папку images
    ys.queriesLikesDia(dbname,path,root)
    ys.queriesDislikesDia(dbname,path,root)
    ys.queriesCommentsDia(dbname,path,root)
    ys.queriesViewsDia(dbname,path,root)
    ys.likesPerViewsDia(dbname,path,root)
    ys.dislikesPerViewsDia(dbname,path,root)
    ys.likesPerDislikeViewsDia(dbname,path,root)
    ys.lastHalfYearDia(dbname,path,root)
    ys.videosPerLastYearDia(dbname,path,root)
    ys.queryTrendsDia(queries,path)
    date=ys.dateConverter(dbname)
    meanLikesViews=ys.getLikesPerViews(dbname,root)
    meanDislikesViews=ys.getDislikesPerViews(dbname,root)
    likesPerDislikes=ys.getLikesPerDislikes(dbname,root)
    lastHalfYear=ys.getLastHalfYear(dbname,root)

    #генерируем html-страничку
ys.htmlGenerator(images,images1,images2,dbname,date,queries,queriesEmbed,list(totalVideos
.values()),root,

list(totalLikes.values()),list(totalDislikes.values()),list(totalComments.values()),
list(totalViews.values()),maxLikeEmbeds,maxDislikeEmbeds,maxCommentsEmbeds,maxViewsEmbeds
,maxLikes,maxDislikes,maxComments,maxViews,meanLikesViews,meanDislikesViews,likesPerDisli
kes,lastHalfYear,3)

    conn.close()

    #print('\nЗавершено')

if __name__ == "__main__":
    youtube_study_analytics()
    html_path = root.split('google_exect')[1]
    html_path = html_path[1 : -1]
    print()
    sys.stdout.buffer.write(html_path.encode())
    sys.stdout = codecs.getwriter("utf-8")(sys.stdout.detach())
```

Приложение Б
(справочное)
Публикация в научном журнале

Журнал «Студенческий вестник» № 2 (100), часть 5, 2020 г.

АНАЛИТИЧЕСКИЙ ИНСТРУМЕНТ ПОИСКА ВОСТРЕБОВАННОГО ОБРАЗОВАТЕЛЬНОГО ВИДЕОКОНТЕНТА В YOUTUBE

Доманов Константин Ильич
студент Вятского государственного университета,
РФ, г. Киров

Макаров Антон Владимирович
студент Вятского государственного университета,
РФ, г. Киров

АННОТАЦИЯ

В статье описано создание инструмента для анализа видеоконтента в Youtube и извлечение полезной информации из исследуемых данных.

Актуальность. В современном образовательном пространстве активно используются информационные технологии. В том числе, их использование обусловлено автоматизированным решением ряда образовательных задач. Например, изучение образовательных потребностей, анализ доступных разработанных онлайн курсов и др. В статье мы рассматриваем аналитический инструмент поиска востребованного образовательного видеоконтента в Youtube. Подобный анализ является вспомогательным инструментам для специалистов по учебно-методической деятельности и позволяет в разы уменьшить время, затраченное на данную деятельность. По этим причинам появилась идея создания информационной системы анализа тематических роликов на платформе Youtube, которая может получать многие статистические данные о популярности видео по определенным запросам, строить графики для сравнения данных, хранить большие объемы информации и формировать подробный HTML-отчет о проведенном анализе.

Реализация и результат. Первым этапом разработки являлось ознакомление с основами и базовыми функциями Youtube API. На данном этапе были освоены особенности выполнения базовых запросов и получения статистики по ним. Для реализации системы был выбран скриптовый язык Python [2, 3] по причине его удобной работы с API и наличием большого количества библиотек для удобного анализа и обработки большого количества данных [1, 4]. При изучении политики сервиса Youtube API были выявлены ограничения в виде квоты на запросы. Каждый день в полночь по Тихоокеанскому времени сервис выдает 10000 единиц квоты для одного API-ключа. Для решения данной проблемы было создано 8 API-ключей с общим количеством квоты 80000 единиц. Данного количества хватало для регулярной отладки в режиме постоянной разработки. При выпуске данного продукта на рынок появится необходимость увеличения квоты через ее покупку.

На втором этапе была разработана база данных с сущностями Query, Video и Result. В Query содержится информация о поисковых запросах. В Result содержится информация об общем количестве видео, лайков, дизлайков, а также об общем количестве комментариев и просмотров. Таблица Video содержит ссылку на конкретное видео, iframe-ссылку для встраивания данного видео в html-отчет, а также название, количество лайков, дизлайков, комментариев, просмотров и дату публикации данного видеоролика. Схема базы данных представлена на рисунке 1.

Рисунок Б.1 – Описание работы программного продукта

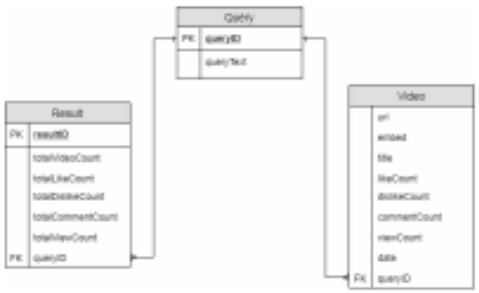

Рисунок 1. Схема базы данных

Также на данном этапе была реализована возможность выполнять запросы по указанным темам из текстового файла по ключевым фразам. Для правильной интерпретации запросов программой необходимо записывать каждый запрос в отдельную строчку.

На следующем этапе было реализовано формирование рабочей директории при запуске скрипта. Данная директория содержит файлы изображений графиков, файл базы данных и HTML-отчет. В имени рабочей директории, файла БД и файла отчета зашивается строка формата «ДЕНЬНЕДЕЛИ_ДД.ММ.ГГГГ_ЧЧ-ММ-СС».

Вся полученная статистическая информация, графики и iframe-ссылки самых популярных видео отображаются в HTML-отчете. Пример HTML-отчета показан на рисунках 2 - 5.

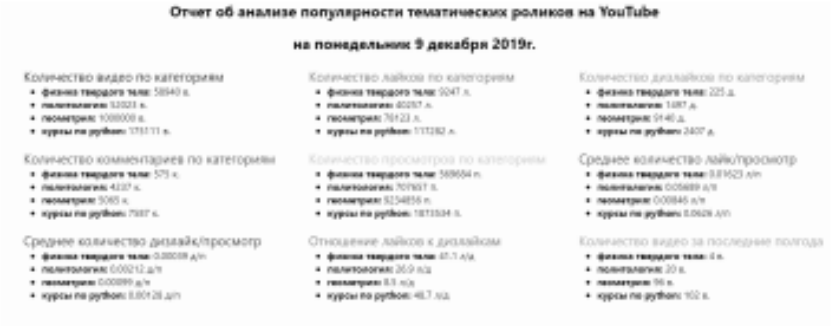

Рисунок 2. Пример HTML-отчета

Рисунок Б.2 – Описание формирования рабочей директории

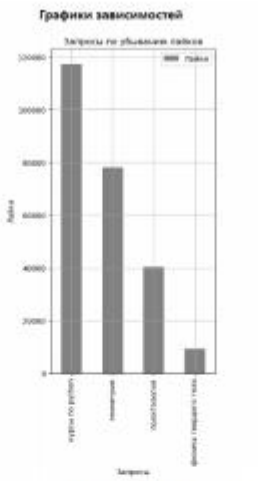

Рисунок 3. Пример HTML-отчета

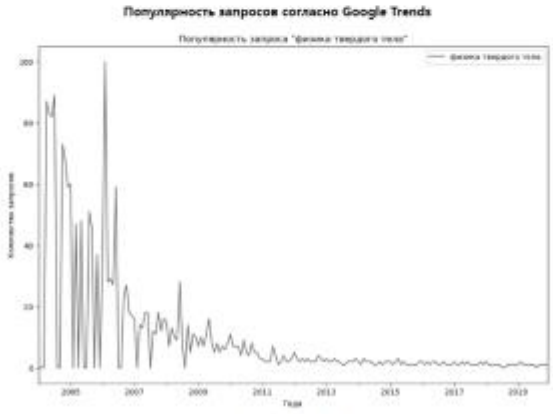

Рисунок 4. Пример HTML-отчета

Рисунок Б.3 – Примеры HTML-отчета

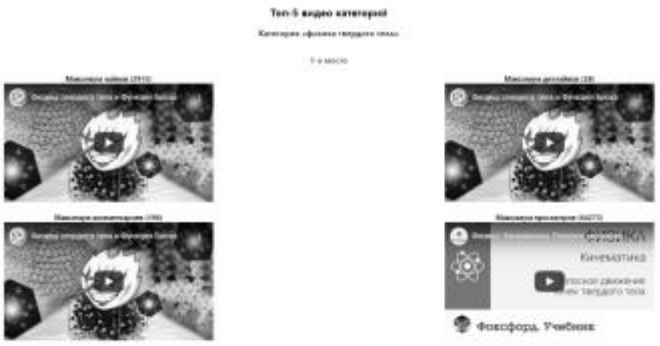

Рисунок 5. Пример HTML-отчета

На следующем этапе была проведена обертка скрипта в исполняемое приложение. Данная операция была выполнена для удобства запуска скрипта на любых компьютерах без установки специального программного обеспечения для его исполнения. В окне приложения можно выбрать готовый текстовый файл с запросами и изменять его при необходимости, либо создать свои запросы в редакторе. Также была добавлена функция выбора лучших видео по каждой из категорий. При начале анализа становится видимым индикатор выполнения, который уведомляет пользователя о проведении расчета. Экранная форма приложения представлена на рисунке 6.

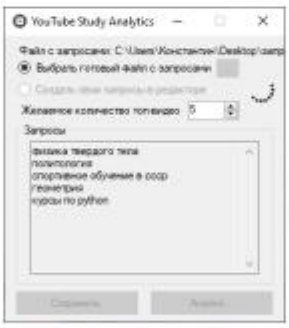

Рисунок 6. Экранная форма

Перспективы. Дальнейшими перспективами развития проекта являются создание расширения для браузера, увеличение количества квоты для использования системы большим количеством пользователей, путем ее покупки, и развитие математических расчетов для более наглядных и точных результатов работы системы.

Вывод: применение разработанной аналитической системы позволит получать необходимую информацию о популярности различных образовательных тем, на основе которой можно делать выводы о составлении образовательных программ и повышать качество образовательного видеоконтента.

Рисунок Б.4 – Подведение итогов

Приложение В
(справочное)
Библиографический список

1. ITtech [Электронный ресурс] // IDEF0. URL: https://itteach.ru/bpwin/metodologiya-idef0 (дата обращения: 10.09.2020).

2. ITtech [Электронный ресурс] // IDEF3. URL: https://itteach.ru/bpwin/metodologiya-idef3 (дата обращения: 21.09.2020).

3. Nazametku [Электронный ресурс] // DFD. URL: https://www.nazametku.com/dlia-raboty/dfd-методология-нотация-принципы-модел/ (дата обращения: 05.10.2020).

4. Analyst [Электронный ресурс] // IDEF1x. URL: http://analyst.by/diagrams/logicheskaya-model-predmetnoy-oblasti (дата обращения: 15.10.2020).

5. ZametkiNaPolyah [Электронный ресурс] // SQLite. URL: https://zametkinapolyah.ru/zametki-o-mysql/chast-5-4-affinirovannye-tipy-dannyx-v-sqlite3.html (дата обращения: 21.10.2020).

6. ZametkiNaPolyah [Электронный ресурс] // SQLite. URL: https://zametkinapolyah.ru/zametki-o-mysql/chast-5-4-affinirovannye-tipy-dannyx-v-sqlite3.html (дата обращения: 15.10.2020).

7. Habr [Электронный ресурс] // UML. URL: https://habr.com/ru/post/458680/ (дата обращения: 01.11.2020).

8. Maccase [Электронный ресурс] // UML Component. URL: https://maccase.ru/android/uml-diagramma-komponentov-opisanie-modelirovanie-na-uml-diagrammy.html (дата обращения: 27.11.2020).