**Содержание**

[Введение 3](#_Toc101371334)

[1 Аналитический раздел 4](#_Toc101371335)

[1.1 Описание предметной области 4](#_Toc101371336)

[1.2 Существующие аналоги 5](#_Toc101371337)

[1.3 Описание бизнес-логики информационной системы 5](#_Toc101371338)

[1.4 Требования к разрабатываемой системе 6](#_Toc101371339)

[1.5 Выводы к разделу 1 8](#_Toc101371340)

[2 Конструкторский раздел 9](#_Toc101371341)

[2.1 Топология системы 9](#_Toc101371342)

[2.2 Описание используемых алгоритмов 11](#_Toc101371343)

[2.3 Диаграмма вариантов использования (Use-Case) 13](#_Toc101371344)

[2.4 Диаграмма последовательности (Sequence) 16](#_Toc101371345)

[2.5 Выводы к разделу 2 17](#_Toc101371346)

[3 Технологический раздел 19](#_Toc101371347)

[3.1 Диаграмма вариантов использования (Use Case) 19](#_Toc101371348)

[3.2 Диаграмма деятельности (Activity) 21](#_Toc101371349)

[3.3 Диаграмма последовательности (Sequence) 23](#_Toc101371350)

[3.4 Диаграмма классов (Class) 24](#_Toc101371351)

[3.5 Диаграмма состояния (State) 26](#_Toc101371352)

[3.6 Диаграмма компонентов (Component) 28](#_Toc101371353)

[3.7 Разработка программного обеспечения 28](#_Toc101371354)

[3.8 Выводы к разделу 3 31](#_Toc101371355)

[Заключение 32](#_Toc101371356)

[Приложение А (обязательное) Листинг программного кода 33](#_Toc101371357)

[manifest.json: 33](#_Toc101371358)

[popup.html: 33](#_Toc101371359)

[httpPost.js: 34](#_Toc101371360)

[100videos.py: 34](#_Toc101371361)

[Приложение Б (справочное) Публикация в научном журнале 39](#_Toc101371362)

[Приложение В (справочное) Библиографический список 43](#_Toc101371363)

Введение

Информационный портал – это интернет-ресурс, предоставляющий пользователям целый комплекс сервисов со сложной разветвленной структурой.

В наше время информационные порталы пользуются большой популярностью во всех сферах жизни общества, которые подвергаются неминуемой автоматизации. Одной из таких сфер является музейный туризм, поскольку удобство покупки билетов в любой интересующий музей определяет потенциальное количество посетителей данного музея.

Таким образом, создание информационного портала, помогающему пользователю купить билет в интересующий его музей, а также получить подробную информацию о музеях и выставках, поможет сэкономить уйму времени, и повысить популярность музейного туризма.

Целью данного курсового проекта является проектирование и разработка информационного портала музеев Российской Федерации.

1. Аналитический раздел

Данный раздел курсовой работы предназначен для того, чтобы более точно понимать направленность решаемой задачи, а именно, разработки информационного портала музеев Российской Федерации. Анализ задачи всегда влечёт за собой определение её требований. Первым делом необходимо произвести описание предметной области, которое поможет сформировать более чёткое понимание разрабатываемого портала.

* 1. Описание предметной области

В настоящее время существует проблема поиска информации в сфере музейной деятельности на территории Российской Федерации. Клиенты не могут находить необходимую информацию о выставках, смотреть часы работы музея и покупать билеты на едином портале.

Некоторые из музеев не имеют своих собственных сайтов. Та часть российских музеев, которые представлены на информационных порталах имеют свою специфику и свои хранилища данных. Поэтому основной задачей является создание единого государственного портала, объединяющего все российские музеи, в том числе и те, у которых нет никакого представительства в интернете. Основной задача данного портала – рассказать обо всех музея Российской Федерации, представить их выставки и постоянные экспозиции, рассказать о часах работы, что будет интересно для туристов и повысить посещаемость музеев.

В настоящий момент существует множество сервисов, предоставляющих информацию только о конкретных музеях, что сильно затрудняет поиск потенциальным посетителям. Ввиду того, что роль автоматизации клиентского сервиса с каждым годом возрастает, появляется необходимость в решении задачи проектирования и разработки системы поиска информации о всех музеев Российской Федерации. К тому же, данная система поможет повысить популярность и посещаемость региональных музеев.

* 1. Существующие аналоги

Среди аналогов можно отметить порталы culture.gosuslugi.ru и museum.ru. Данный проект должен иметь следующие преимущества перед существующими аналогами:

* Поиск информации о музеях не только в больших городах, но и в регионах.
* Поиск актуальной информации о выставках и точном местоположении музеев.
* Возможность покупки билета в музей на портале.
* Удобный интерфейс и высокая скорость загрузки страниц портала.
  1. Описание бизнес-логики информационной системы

Проект должен представлять собой портал для поиска информации и покупки билетов в музеи. В данной информационной системе предполагается наличие двух типов пользователей: посетитель и администратор.

Потенциальный посетитель регистрируется на портале и указывает информацию о себе: имя, фамилия, логин.

Администратор ответственен за добавление музеев на портал. При добавлении нового музея, он должен указать всю информацию о добавляемой организации: название музея, его описание, адрес местонахождения, наименование юридического лица, ИНН, ОГРН, тип музея, электронную почту.

После успешного добавления музея, появляется базовая страничка с главной информацией о нем, в которую он может вносить правки, а также добавлять информацию о проходящих выставках.

При добавлении информации о выставках, администратор должен указывать название, краткое описание выставки, информацию о билетах, даты проведения (в случае временной выставки).

На основе этой информации клиенты, посещающие портал, производят поиск подходящих музеев, узнают их особенности, а также имеют возможность приобрести билеты на представленные на портале выставки.

Помимо этого, зарегистрированный пользователь должен иметь возможность просмотра своего профиля, купленных им билетов и менять цветовую тему портала.

Администратор имеет возможности удаления музеев, выставок, а также просмотра статистики о регистрации пользователей и поступлении средств после покупки билетов на счет музеев.

* 1. Требования к разрабатываемой системе

1. Система должна обеспечивать разделение пользователей на две роли:

* клиент;
* администратор;

1. Каждый музей, представленный на портале, должен быть классифицирован по типу его деятельности. Система должна обеспечивать музею выбор категории из представленного ниже списка:

* архитектурно-ансамблевый;
* естественнонаучный;
* краеведческий;
* исторический;
* художественный;
* научно-технический;
* литературный;
* театральный;
* музыкальный;
* музей-заповедник.

1. В случае недоступности некритичного функционала, должна осуществляться деградация функциональности.
2. Система должна обеспечивать валидацию вводимых данных через интерфейс приложения при операциях добавления, изменения контента портала, а также при аутентификации и регистрации.
3. Система должна обеспечивать регистрацию.
4. Система должна обеспечивать авторизацию и аутентификацию пользователей.
5. Система должна предоставлять **клиенту** следующие функции:

* просмотр информации о представленных музеях: тип музея, описание, местоположение, контактные данные, представленные в нем выставки;
* покупка билетов на выставки;
* просмотр профиля;
* просмотр истории купленных билетов;
* смена цветовой палитры портала.

1. Система должна предоставлять **администратору** следующие функции:

* добавление информации о музеях;
* изменение информации о музеях;
* удаление информации о музеях;
* добавление информации о проходящих выставках;
* удаление информации о выставках музея;
* возможность просмотра статистики поступлений денежных средств на счет музеев;
* возможность просмотра статистики о регистрации новых пользователей;
* просмотр профиля;
* смена цветовой палитры портала.

1. Каждый сервис разрабатываемой системы при необходимости может иметь доступ к связанной с ним базе данных, но не должен иметь доступа к базам данных других сервисов.
2. Все запросы между сервисами требуют авторизации.
   1. Выводы к разделу 1

В представленном разделе произведен анализ предметной области и существующих аналогов. Были определены основные требования, которым должна удовлетворять разрабатываемая система. Так же в данном разделе было произведено подробное описание бизнес-логики данного информационного портала.

1. Конструкторский раздел

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 будет описана архитектура и алгоритмы разрабатываемой системы, описаны отдельные компоненты системы, а также спроектированы основные диаграммы, описывающие работу системы и взаимодействие отдельных компонентов.

* 1. Топология системы

Система будет состоять из фронтэнда и пяти сервисов, что наиболее целесообразно для реализации ее основного назначения. Топология разрабатываемой системы представлена на рисунке 2.1.

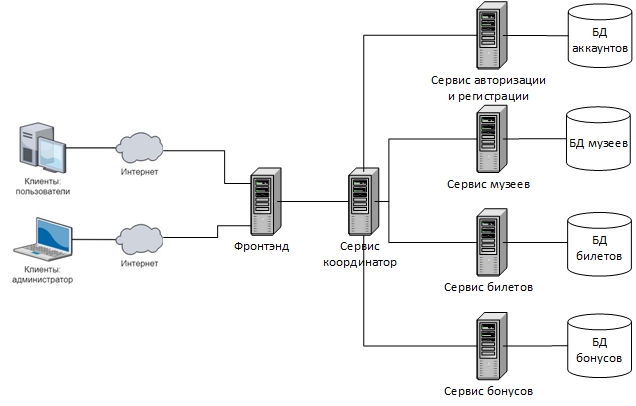


Рисунок 2.1 – Топология системы

**Сервис авторизации и регистрации** отвечает за пользователей портала и реализует следующие функции:

* + регистрация пользователя;
  + аутентификация пользователя;
  + просмотр профиля;
  + авторизация пользователя;
  + валидация токена;
  + получение и изменение цветовой темы портала.

**Сервис музеев** отвечает за хранение информации о представленных на портале музеях и реализует следующие функции:

* + получение списка музеев с условиями фильтрации по типу и местоположению;
  + получение информации о конкретном музее;
  + получение информации о проходящий в музее выставках и постоянных экспозициях;
  + добавление нового музея;
  + изменение информации конкретного музея;
  + удаление музея;
  + добавление выставки;
  + удаление выставки;
  + получение списка всех музейных типов;
  + начисление на счет музея средств после продажи билета.

**Сервис билетов** отвечает за хранение информации о билетах и реализует следующие функции:

* + покупка билета на конкретную экспозицию или выставку музея;
  + просмотр истории купленных пользователем билетов.

**Сервис статистики** отвечает за хранения информации о статистике и реализует следующие функции:

* + добавление и получение информации о всех поступлениях на счет музеев;
  + добавление и получение информации о новых пользователях системы.

**Сервис координатор** отвечает за диспетчеризацию запросов и предоставляет оптимальный унифицированный API.

**Фронтэнд** принимает запросы от пользователей по протоколу HTTP и анализирует их. На основе проведенного анализа фронтэнд выполняет запросы к координационному сервису, получает ответы и отсылает их пользователю.

* 1. Описание используемых алгоритмов

Все сервисы разрабатываемого портала используют протокол HTTP для получения и отправления информации. Все запросы, кроме получения токена проходят через координационный сервис.

Аутентификация на портале реализована с помощью Json Web Token. Json Web Token (JWT) — это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности.

Типичный алгоритм аутентификации на основе токенов представлен на рисунке 2.2.

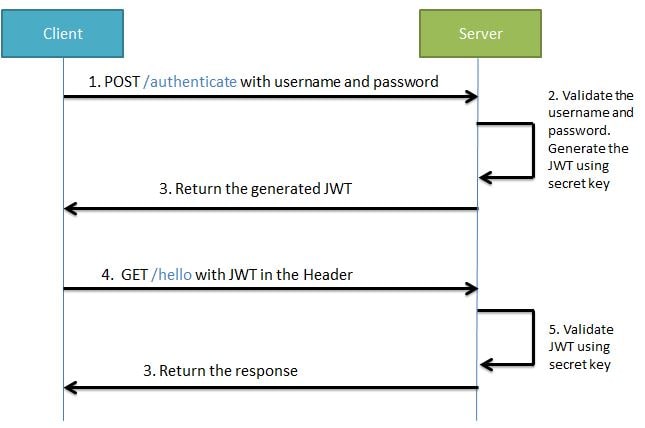


Рисунок 2.1 – Аутентификация на основе токенов

В разрабатываемой системе аутентификация на основе JWT-токена реализована следующим образом:

* + - 1. При успешной авторизации или регистрации нового пользователя, сервис авторизации и регистрации генерирует токен сроком действия в 10 минут и отправляет его на фронтэнд, который в свою очередь записывает токен в куки.
      2. При выполнении запросов, требующих авторизации, json web token подставляется в заголовок запроса. Сервис координатор перед выполнением запроса валидирует токен на сервисе авторизации и регистрации.
      3. Валидация токена осуществляется путем проверки даты действия токена и совпадения исходной уникальной подписи с тестовой. Если тестовая подпись совпадает с исходной, то это означает, что полезная нагрузка и заголовок не были изменены. Для генерации уникальной подписи используется секрет.
      4. В случае валидности токена, сервис координатор продолжает успешное выполнения запроса. В случае невалидности токена, сервис координатор возвращает 401 код ошибки HTTP.

Помимо этого, из важных особенностей реализации стоит отметить, что каждый сервис, кроме сервиса координатора имеет свое собственное хранилище данных, доступ к которому есть только у него.

База данных сервиса авторизации и регистрации хранит пользовательские пароли в хэшированном виде. В качестве алгоритма хэширования используется bcrypt.

bcrypt — адаптивная криптографическая хэш-функция формирования ключа, используемая для защищенного хранения паролей. Функция основана на шифре Blowfish. Для защиты от атак с помощью радужных таблиц, bcrypt использует соль, кроме того, функция является адаптивной, время её работы легко настраивается и её можно замедлить, чтобы усложнить атаку перебором.

* 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.

Разработанные диаграммы вариантов использования для актеров клиент и администратор представлены на рисунках представлены на рисунках 2.2 и 2.3 соответственно.



Рисунок 2.2 – Диаграмма вариантов использования для клиента



Рисунок 2.3 – Диаграмма вариантов использования для администратора

* 1. Диаграмма последовательности (Sequence)

Диаграммы последовательности действий отображают взаимодействие объектов, упорядоченное во времени.

К основным компонентам диаграммы последовательности можно отнести:

* Объекты – основные компоненты системы.
* 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.

В ходе проектирования информационного портала музеев Российской Федерации была построена диаграмма последовательности системы, которая представлена на рисунке 2.4.

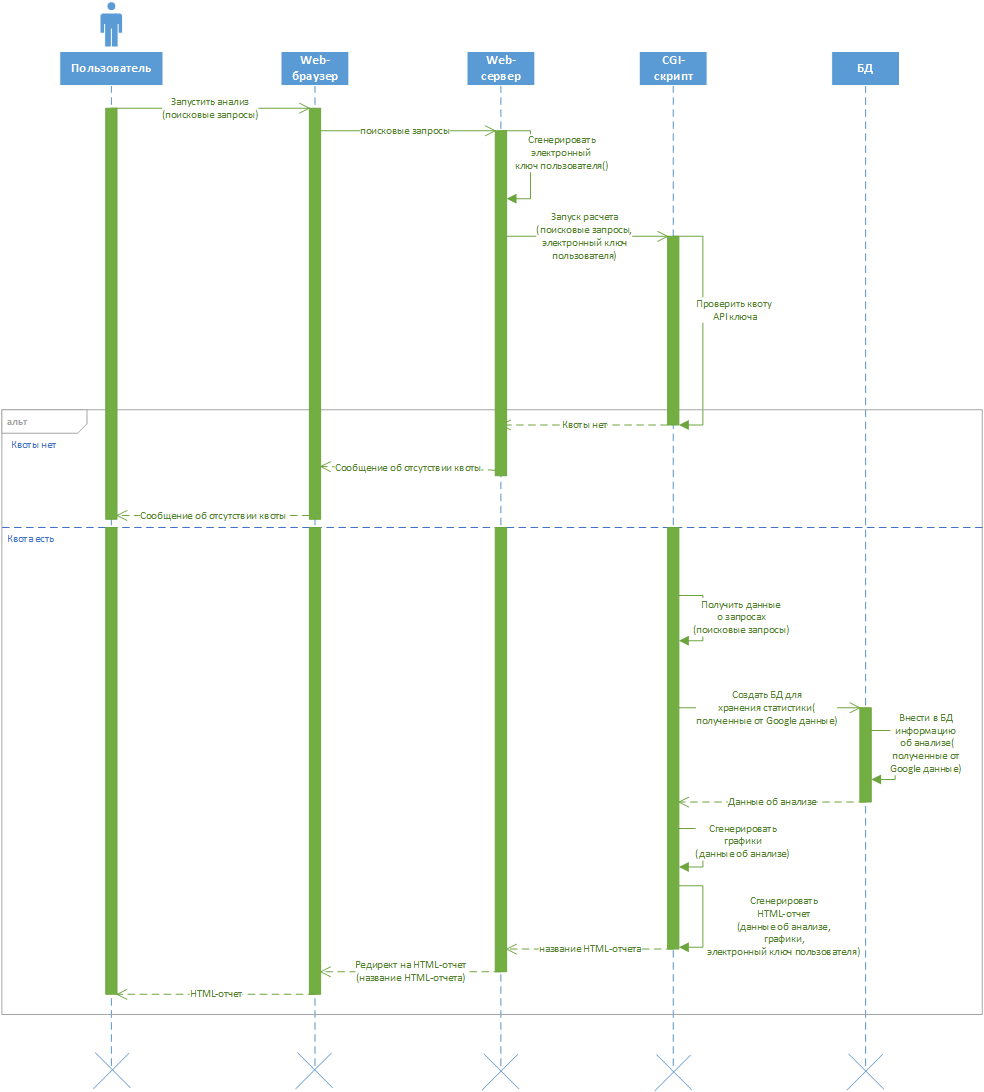


Рисунок 3.3.1 – Диаграмма последовательности системы

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

* 1. Выводы к разделу 2

В рассмотренном выше разделе описана архитектура, основные алгоритмы разрабатываемой системы и отдельные компоненты информационного портала.

Помимо этого, были спроектированы диаграммы вариантов использования для каждой из представленных на портале ролей. А также диаграммы последовательности действий, показывающие взаимодействие микросервисов.

1. Технологический раздел

На данном этапе разработки информационного портала произведено описание типов и структур данных в нотации IDEF1x, а также тестирование и поведение системы в случае отказа.

* 1. Структура базы данных

Модели данных служат для проектирования структуры постоянных хранилищ данных, используемых системой. При проектировании информационного портала всех музеев Российской Федерации были разработаны модели данных сервиса авторизации и регистрации, сервиса музеев, сервиса билетов и сервиса статистики в нотации IDEF1x.

Диаграммы базы данных сервиса авторизации и регистрации, сервиса музеев, сервиса билетов и сервиса статистики представлены на рисунках 3.1 – 3.4 соответственно.

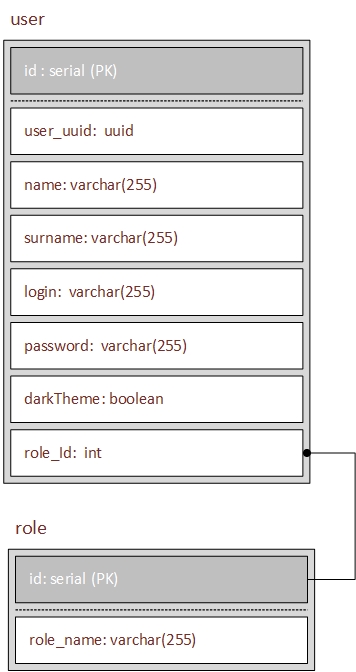


Рисунок 3.1 – БД сервиса авторизации и регистрации

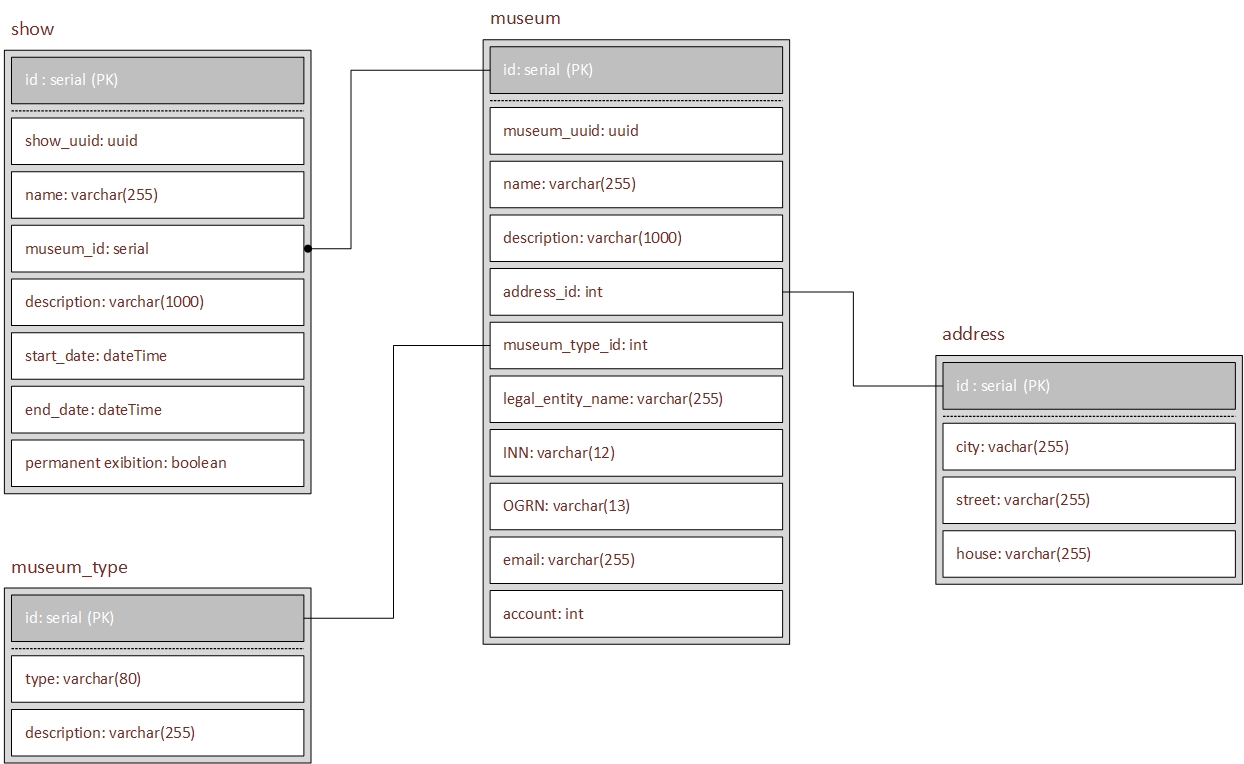


Рисунок 3.2 – БД сервиса музеев

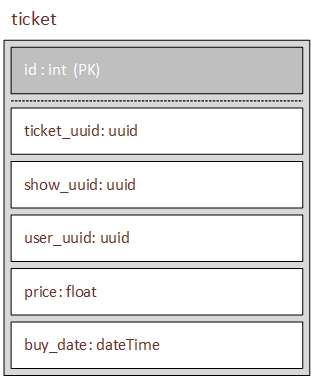


Рисунок 3.3 – БД сервиса билетов

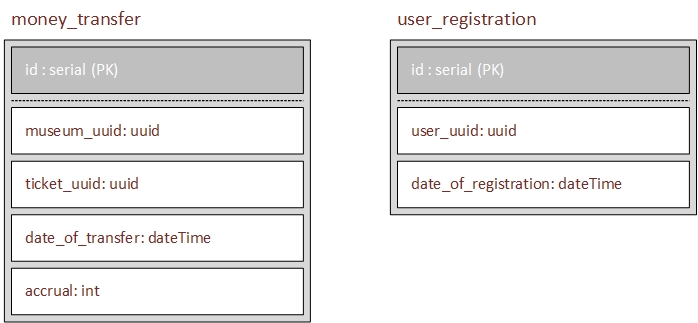


Рисунок 3.4 – БД сервиса статистики

* 1. Тестирование

При разработке информационного портала всех музеев Российской Федерации необходимо учитывать случаи отказа отдельных компонентов системы. В случае ошибки или недоступности некритичного функционала должна производиться деградация функциональности.

В данном проекте деградация функциональности реализована при запросах к сервису статистики. В случае, если при регистрации нового пользователя сервис статистики не отвечает, то пользователь получает ответ об успешной регистрации, а сервис координатор ставит запрос на добавление статистики о зарегистрированных пользователях в очередь. После чего, сервис координатор каждые 5 секунд выполняет запрос из очереди. В случае неудачи, запрос снова помещается в очередь.

Также, деградация функциональности реализована и при покупке билета на выставку. В случае, если при покупке билета сервис статистики не отвечает, то пользователь получает ответ об успешном приобретении билета, а сервис координатор ставит запрос на добавление истории начислений на счет музея в очередь. После чего, сервис координатор каждые 5 секунд выполняет запрос из очереди. В случае неудачно запрос снова помещается в очередь.

Для выполнения запросов, создается отдельный поток для каждой очереди.

Помимо этого, в разрабатываемой системе были предусмотрены случаи ошибок при получении информации.

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.

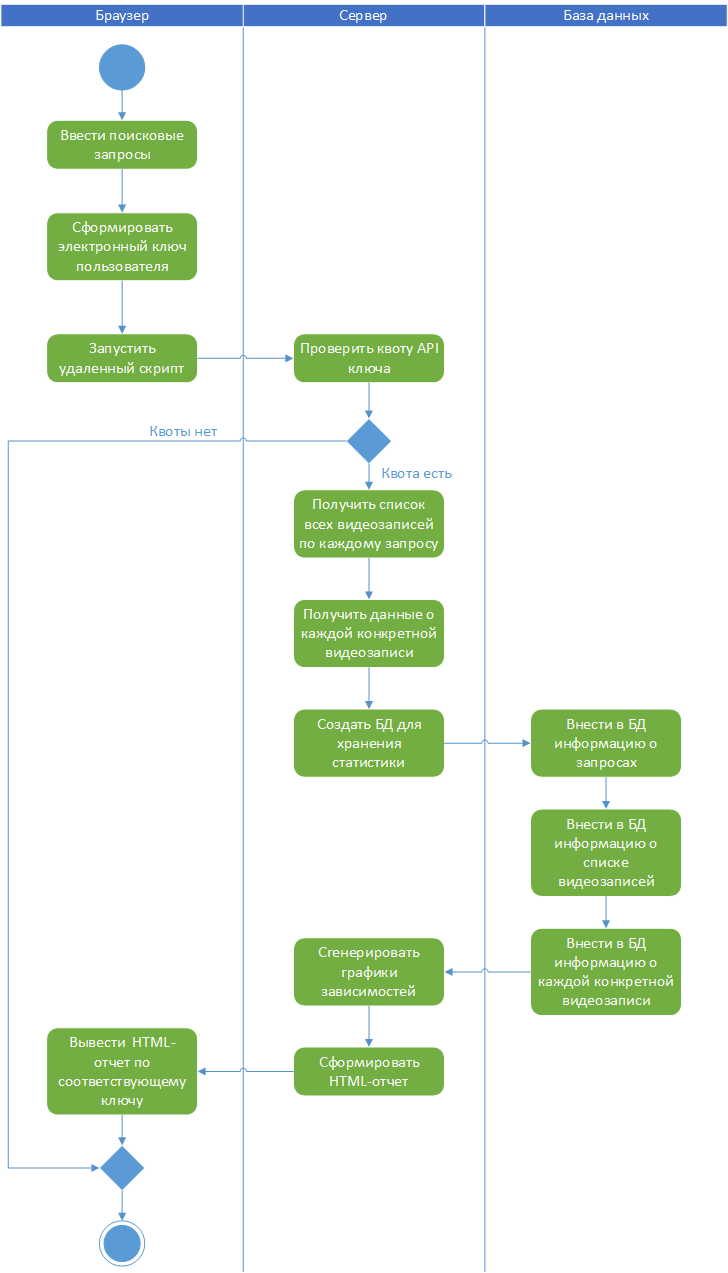


Рисунок 3.2.1 – Диаграмма деятельности

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.

* 1. 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.

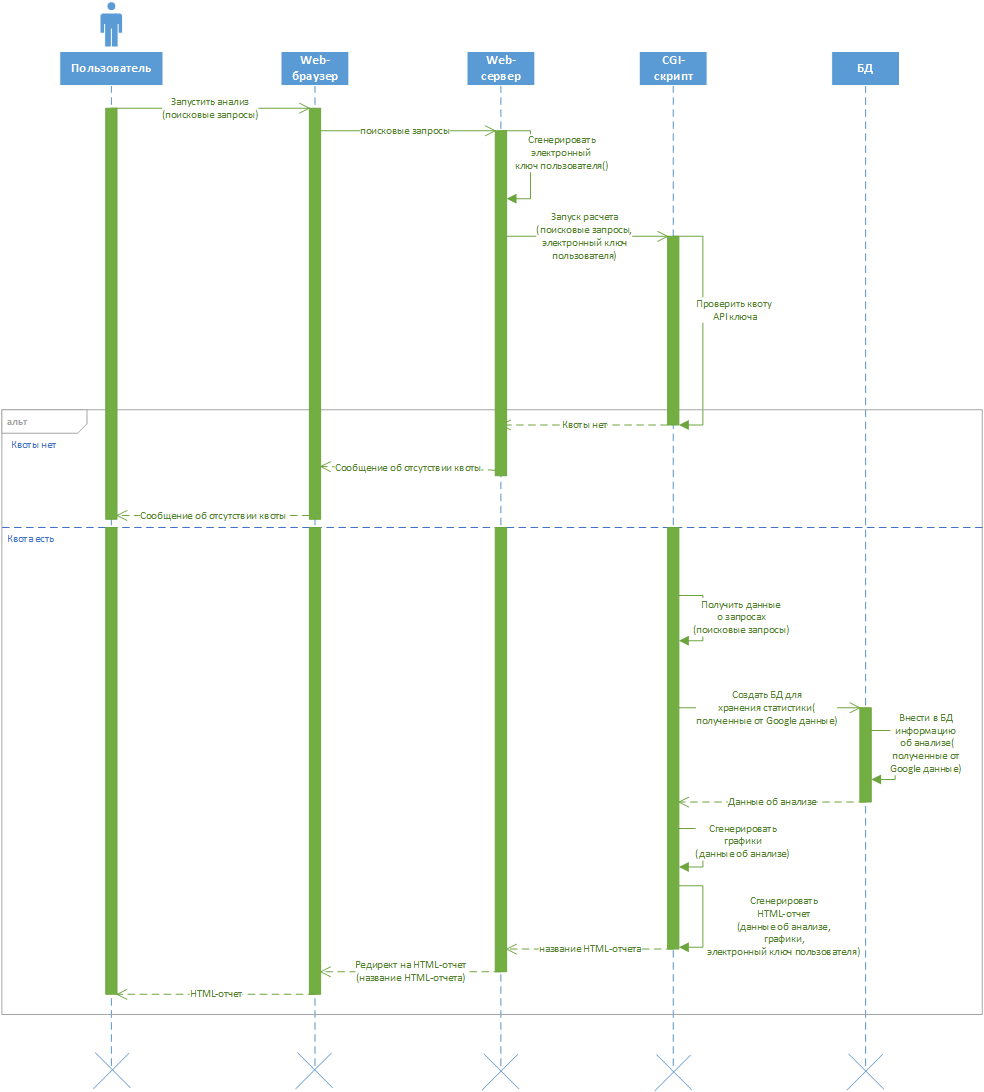


Рисунок 3.3.1 – Диаграмма последовательности системы

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

* 1. 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.

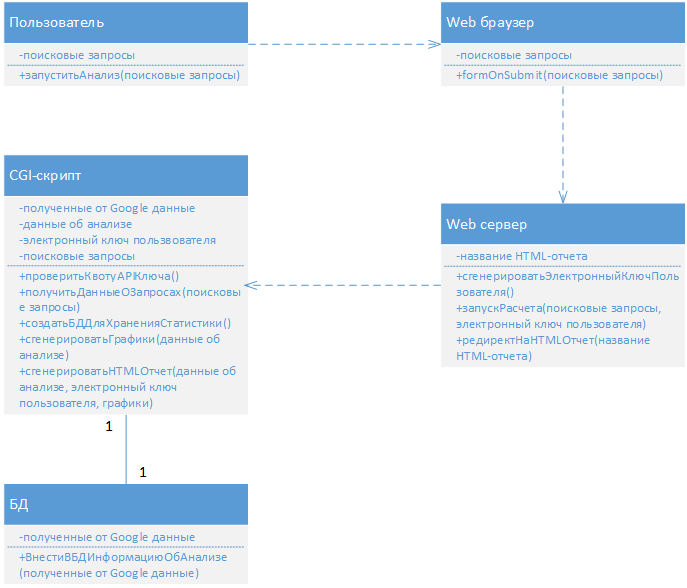


Рисунок 3.4.1 – Диаграмма классов

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

* 1. 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.

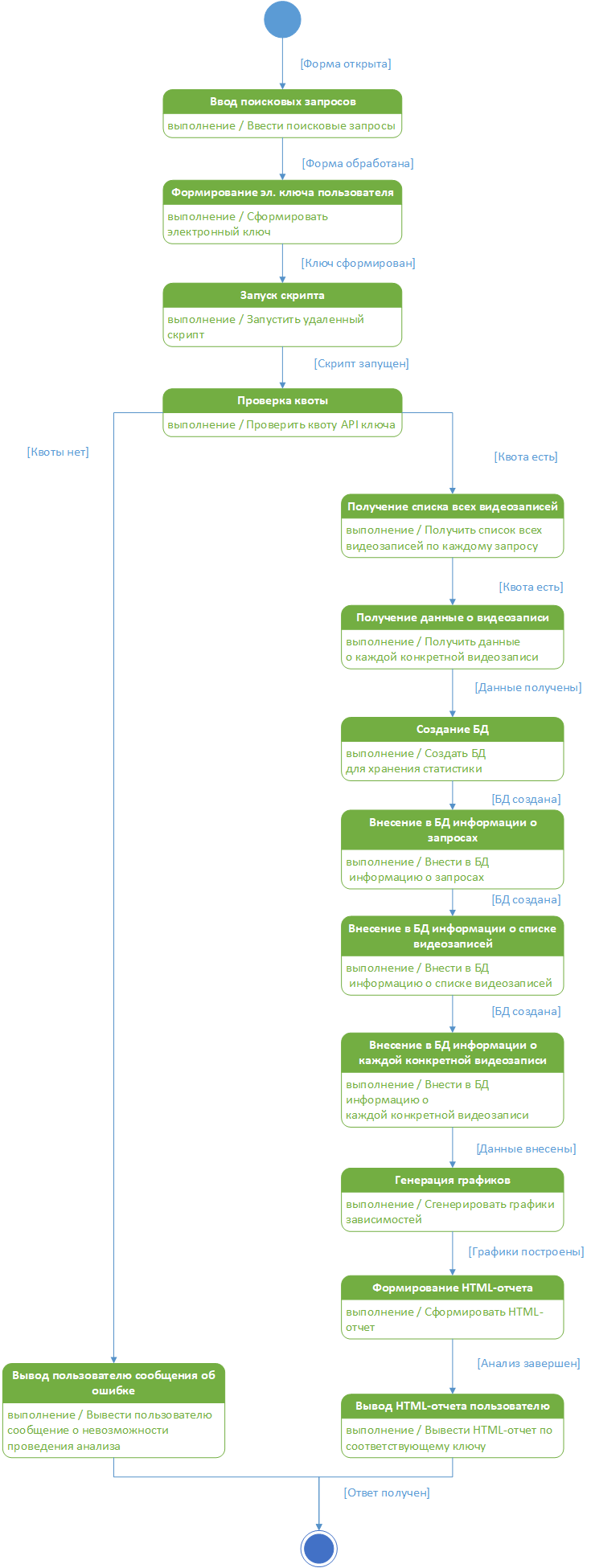


Рисунок 3.5.1 – Диаграмма состояний

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

* 1. 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.

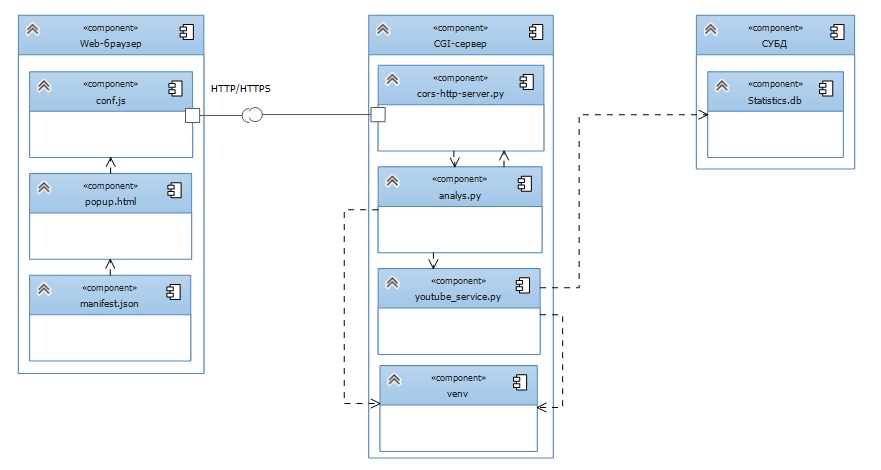


Рисунок 3.6.1 – Диаграмма компонентов

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

* 1. 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.

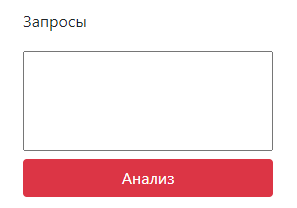


Рисунок 3.7.1 – Экранная форма взаимодействия с пользователем

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.



Рисунок 3.7.2 – Основные характеристики запросов

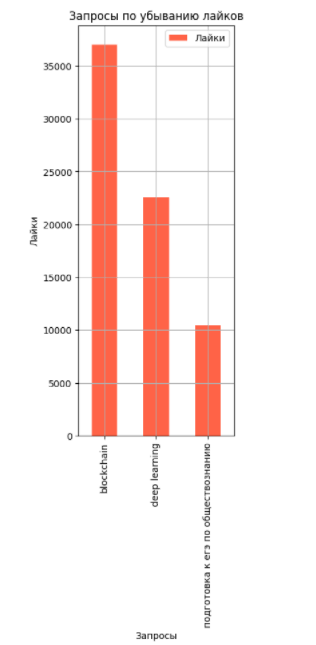


Рисунок 3.7.3 – График запросов по убыванию лайков

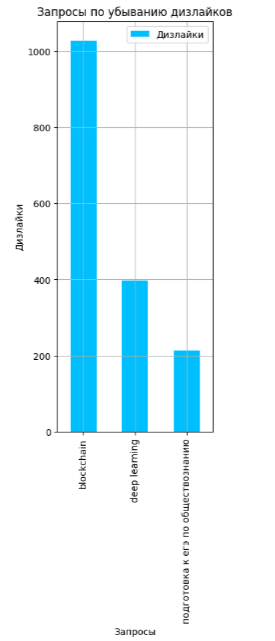


Рисунок 3.7.4 – График запросов по убыванию дизлайков

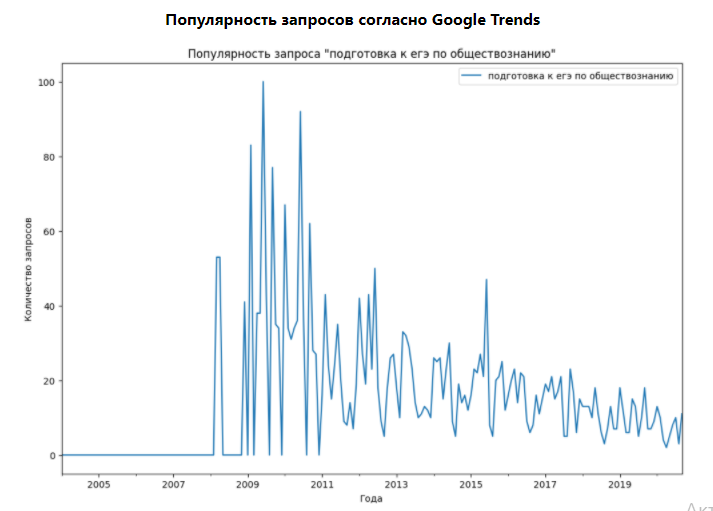


Рисунок 3.7.5 – График популярности запроса «подготовка к егэ по обществознанию»

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.

* 1. 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,likesPerDislikes,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

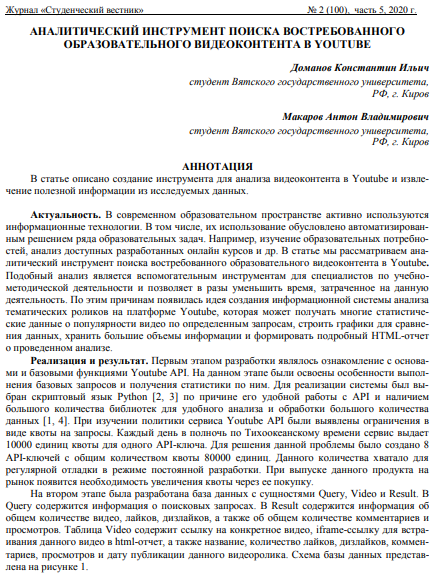
****

Рисунок Б.1 – Описание работы программного продукта

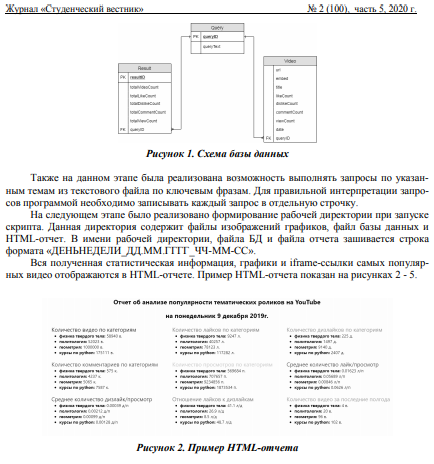
****

Рисунок Б.2 – Описание формирования рабочей директории

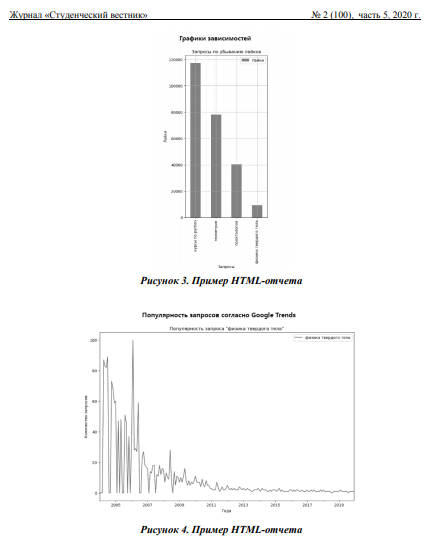
****

Рисунок Б.3 – Примеры HTML-отчета

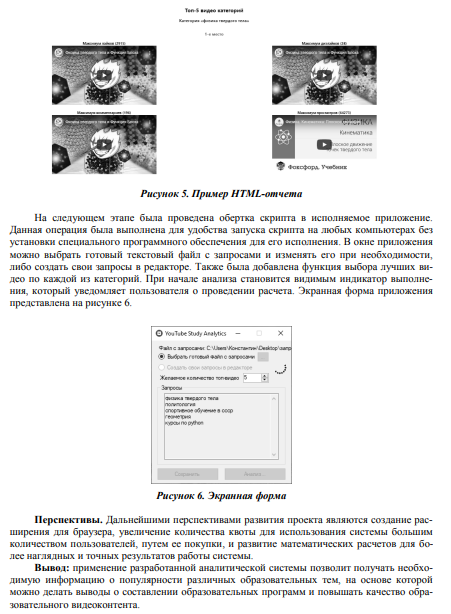
****

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).