

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт** **информационных систем** **и технологий** | **Кафедра**  **информационных систем** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Сычева Анастасия Антоновна** | | | |
| **Тема: «Разработка программного и информационного обеспечения мультимедийной картотеки с веб-доступом»»** | | | |
| **Выпускная квалификационная работа на присвоение квалификации «бакалавр»  по направлению 09.03.02 «Информационные системы и технологии»** | | | |
|  | | | |
|  | | Регистрационный № \_\_\_\_\_\_\_ | |
| **Заведующий кафедрой**  **д.т.н., проф.** |  | | **Позднеев Б.М.** |
|  | подпись | |  |
| Руководитель **Ст. преп.** |  | | **Овчинников П.Е.** |
|  | подпись | |  |
| **Студент группы ИДБ-13-12** |  | | **Сычева А.А.** |
|  | подпись | |  |

Москва 2017 г.

**Оглавление**

[ВВЕДЕНИЕ 3](#_Toc483433787)

[ГЛАВА 1. Анализ состояния вопроса и обоснование цели и задач ВКР 4](#_Toc483433788)

[1.1. Постановка задачи 4](#_Toc483433789)

[1.2. Нормативно-правовая база 7](#_Toc483433790)

[1.3. Средства моделирования 10](#_Toc483433791)

[1.4. Средства разработки 12](#_Toc483433792)

[1.4.1. HTML. CSS 12](#_Toc483433793)

[1.4.2. JavaScript 14](#_Toc483433794)

[1.4.3. PHP 15](#_Toc483433795)

[1.4.4. PHPMyAdmin 16](#_Toc483433796)

[ГЛАВА 2. Разработка комплекса моделей информационной системы 18](#_Toc483433797)

[2.1. Функциональная модель (IDEF0) 19](#_Toc483433798)

[2.2. Модель потоков данных (DFD) 24](#_Toc483433799)

[2.3. Реляционная модель базы данных (ERD) 29](#_Toc483433800)

[ГЛАВА 3. Разработка и исследовательские испытания прототипа программного и информационного обеспечения 32](#_Toc483433801)

[3.1. Разработка прототипа 32](#_Toc483433802)

[3.2. Испытания прототипа 42](#_Toc483433803)

[ЗАКЛЮЧЕНИЕ 47](#_Toc483433804)

[Список источников 49](#_Toc483433805)

[ПРИЛОЖЕНИЕ А 51](#_Toc483433806)

[ПРИЛОЖЕНИЕ Б 58](#_Toc483433807)

[ПРИЛОЖЕНИЕ В 62](#_Toc483433808)

ВВЕДЕНИЕ

На данном этапе развития общества всю большую роль в жизни занимают различные информационные технологии. Также в последнее время прослеживается четкая тенденция повышения популярности веб-ресурсов и облачных сервисов – работа с информацией переносится с рабочей машины пользователя в глобальную сеть Интернет. Поэтому именно сейчас работы в области разработки онлайн-ресурсов являются актуальными и востребованными.

Веб-приложения и интернет сайты используются в самых различных сферах жизни общества. «Всемирная паутина» становится не только площадкой для получения информации и общения, но и инструментом реализации бытовых нужд: таких как совершение покупок, оплата коммунальных услуг и др. Любая уважающая себя организация, будь то учебное заведение или коммерческий холдинг, имеет свой веб-сайт, многие торговые сети и магазины поддерживают свой интернет-магазин. Свои веб-странички имеют и частные предприниматели, музыкальные группы, даже рядовые пользователи. Огромное количество самой разнообразной информации на данный момент хранится и обрабатывается в глобальном информационном пространстве.

Широкое распространение как в «настольных» приложениях, так и в различных веб-ресурсах получили реляционные базы данных, так как являются одним из самых удобных способов хранения информации. Базы данных необходимы для многих областей деятельности человека, будь то банковское дело, продовольственный магазин или учет домашних расходов.

Можно сделать вывод, что на данном этапе развития общества с учетом столь стремительного развития сети Интернет, разработка веб-приложений и сайтов, способных работать с базами данных, является актуальным и востребованным процессом во всех предметных областях, например, хранение мультимедийных данных и мета информации о них.

ГЛАВА 1. Анализ состояния вопроса и обоснование цели и задач ВКР

В качестве цели выполнения выпускной квалификационной работы было выбрано обеспечение технологической поддержки процессов формирования и использования мультимедийной картотеки в повседневной, учебной и профессиональной деятельности различных групп пользователей.

В качестве предметной области данной работы была выбрана разработка мультимедийной распределенной картотеки, в частном случае – музыкальной картотеки, что обусловлено широким распространением в наше время музыки и саундтреков. Данная разработка призвана помочь пользователю составить список всех важных для него музыкальных композиций, с прикреплением необходимой о них информации, что поможет классифицировать и упорядочить свою собственную музыкальную медиатеку пользователя. Подразумевается, что разработанный программный продукт станет полезным инструментом не только для меломана, но и для человека, не сильно увлекающегося музыкой.

1.1. Постановка задачи

В процессе выполнения данной работы должна быть выполнена разработка и обеспечение технологической поддержки процессов формирования и использования мультимедийной картотеки, т.е. разработка и поддержка прототипа программного продукта. Разрабатываемая система должна быть способна хранить в себе всю необходимую информацию о музыкальных композициях, предоставлять возможность прав доступа на изменение и отображение информации, поддерживать свою систему авторизации. На этапе проектирования разрабатываемой системы необходимо разработать функциональные модели в формате IDEF0, моделей потоков данных в формате DFD, моделей базы данных в формате ERD.

В силу широкого использования сети Интернет было принято решений вести разработку в веб среде, т.е. написание веб-приложения (веб сайта).

Веб-сайт представляет собой объединенные общей тематикой и дизайном несколько веб-страниц, хранящихся на веб-сервере. Каждый веб-сайт имеет имя, т.е. адрес в сети интернет. Для отображения вею-страниц пользователи используют специальные программы, называемые веб-браузерами.

Веб-страница в свою очередь является гипертекстовым ресурсом Всемирной паутины, способная содержать ссылки для перехода на другие страницы, изображения, медиафайлы и т.п. Чаще всего такие страницы имеют расширения .html, .htm, .php.

 Классифицируя сайты по используемым технологиям, можно выявить три группы:

  1) Статические сайты - сделанные по классической технологии html, написанные полностью на языке хранящиеся на сервере в неизменном вид.

  2) Динамическиесайты - формируются при помощи скриптов, которые могут быть написаны на различных языках. Содержание и вид динамической страницы с одним и тем же адресом может быть совершенно различным в зависимости от разных.

  3) Флэш-сайты- разработанные при помощи технология флэш.

Разрабатываемый веб-сайт относится ко второй группе – динамическим, отображаемое содержимое страницы сайта не хранится на сервере в неизменном виде, а меняется в зависимости от различных условий.

Такой сайт предоставляет пользователю интерактивность – способность системы активно и [адекватно](https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%B5%D0%BA%D0%B2%D0%B0%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8) реагировать на действия пользователя. Подобный подход к разработке идеально подходит в условиях работы сайта с базами данных. База данных позволяет упорядочить информацию по различным признакам и быстро производить ее выборку. При этом очень важно выбрать правильную модель данных.

База данных – это информационная модель, позволяющая упорядоченно хранить данные о группе объектов, обладающих одинаковым набором свойств. В базах данных типы организации данных и связей между ними можно в общем разделить на три группы:

1. Иерархический (в виде дерева) - существует упорядоченность элементов, при этом один элемент считается главным, остальные – подчиненными (см. рис. 1.1.1).

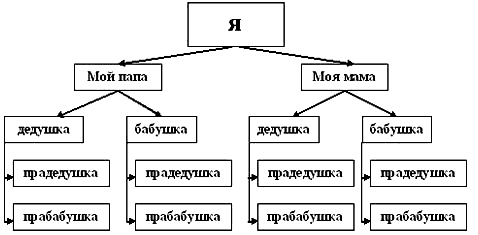


Рис. 1.1.1. Иерархическая база данных.

1. Сетевой - существует возможность устанавливать дополнительно к вертикальным иерархическим связям горизонтальные связи (см. рис. 1.1.2).

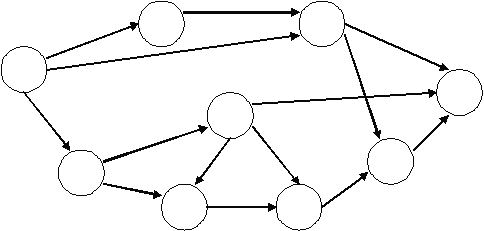


Рис. 1.1.2. Сетевая база данных.

1. Реляционный - совокупность взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного типа (см. рис. 1.1.3).

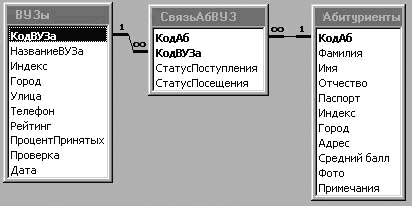


Рис. 1.1.3. Реляционная база данных.

Разрабатываемая в рамках данной работы база данных будет принадлежать к третьей группе – реляционным базам данных, так как именно эта группа обладает такими преимуществами, как наглядность и понятность, а также скорость поиска нужной информации.

1.2. Нормативно-правовая база

В условиях рыночной экономики успех фирмы напрямую зависит от степени удовлетворенности потребителей, а также ожидаемым или непредусмотренным влиянием на другие заинтересованные стороны, что определяет качество продуктов и услуг.

ГОСТ Р ИСО 9001-2015 «Системы менеджмента качества. Требования» устанавливает требования к системе менеджмента качества. Применение системы менеджмента качества является стратегическим решением для организации, которое может помочь улучшить результаты ее деятельности и обеспечить прочную основу для инициатив, ориентированных на устойчивое развитие [2].

В рамках данного проекта необходимо обеспечить высокий уровень качества взаимодействия системы с пользователем, что включает в себя как безотказность, корректность и быстродействие выполнения пользовательских запросов, так и удобство пользования, и оформление графического интерфейса.

ГОСТ 34.003-90 «Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения» устанавливает термины и определения основных понятий в области автоматизированных систем (АС) и распространяется на АС, используемые в различных сферах деятельности (управление, исследования, проектирование и т.п., включая их сочетание), содержанием которых является переработка информации [3].

Разрабатываемый программный продукт является автоматизированной информационной системой (АИС). Такие функции, как поиск и изменение информации в базе данных выполняется самой системой по введенным входным данным. Пользователю (в том числе администратору) нет необходимости знать и понимать структуру системы. Для взаимодействия с разрабатываемым веб-приложением достаточно графического интерфейса.

К данной выпускной квалификационной работе применимы следующие термины, установленные приведенным стандартом [3]:

* автоматизированная система (АС) - состоит из персонала и комплекса средств автоматизации его деятельности, реализует информационную технологию выполнения установленных функций;
* методическое обеспечение автоматизированной системы - совокупность документов, описывающих технологию функционирования АС, методы выбора и применения пользователями технологических приемов для получения конкретных результатов при функционировании АС;
* программное обеспечение автоматизированной системы - совокупность программ на носителях данных и программных документов, предназначенная для отладки, функционирования и проверки работоспособности АС;
* сопровождение автоматизированной системы - деятельность по оказанию услуг, необходимых для обеспечения устойчивого функционирования или развития АС;
* входная информация автоматизированной системы – информация, поступающая в АС в виде документов, сообщений, данных, сигналов, необходимая для выполнения функций АС;
* выходная информация автоматизированной системы - информация, получаемая в результате выполнения функций АС и выдаваемая на объект ее деятельности, пользователю или в другие системы;
* диалоговый режим выполнения функции автоматизированной системы - режим выполнения функции АС, при котором человек управляет решением задачи, изменяя ее условия и (или) порядок функционирования АС на основе оценки информации, представляемой ему техническими средствами АС.

В стандарте ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств» при помощи устоявшейся терминологии устанавливается общая структура про­цессов жизненного цикла программных средств, на которую можно ориентироваться в программной инду­стрии. Однако стандарт не требует использования какой-либо конкретной модели жизненного цикла или какой-либо заданной сово­купности стадий.

Жизненный цикл (life cycle) - развитие системы, продукта, услуги, проекта или других изготов­ленных человеком объектов, начиная со стадии разработки концепции и заканчивая прекращением приме­нения [4].

Модель жизненного цикла (life cycle model) - структура процессов и действий, связанных с жизненным циклом, организуемых в стадии, которые также служат в качестве общей ссылки для установ­ления связей и взаимопонимания сторон [4].

В процессе выполнения данной выпускной квалификационной работы предполагается использование следующих процессов: процессы планирования проекта, оценки проекта, менеджмента решений, менеджмента информации, анализа системных требований, проектирования архитектуры системы, реализации, комплексирования системы, квалификационного тестирования системы, функционирования программных средств, сопровождения программных средств [4].

Часть 3 ГОСТ Р 56923-2016/ISO/IEC TR 24748-3:2011 «Информационные технологии. Системная и программная инженерия. Управление жизненным циклом» является руководством по применению ИСО/МЭК 12207 (Процессы жизненного цикла программных средств), т.е. руководством по управлению жизненным циклом в области программных систем [5].

Под процессом подразумевается совокупность действий, преобразующих входы в выходы. Процесс может управляться с помощью инструкций, директив организации, законов и т.п. У любого процесса есть механизмы – необходимые для его выполнения ресурсы, рабочая сила, технологии и инструменты. Входы – входные данные процессов, могут поступать как извне проекта и организации, так и из других процессов. Выходы – результаты выполнения процессов.

Согласно данному стандарту необходимо определить входы и выходы процессов, а также обеспечивающие механизмы и управление.

ГОСТ 24026-80 «Исследовательские испытания. Планирование эксперимента. Термины и определения» устанавливает термины и определения основных понятий в области исследовательских испытаний, относящихся к разделу планирования эксперимента [6].

В рамках разрабатываемого прототипа программного продукта необходимо провести испытания таких процессов, как получение данных запроса от пользователя, его выполнение и вывод результата (в том числе и авторизация).

1.3. Средства моделирования

Под моделированием понимается построение представления изучаемого объекта, отражающего наиболее важные в ходе конкретного исследования черты и свойства объекта, а также изучение уже существующих моделей. В рамках выполнения данной работы появляется необходимость построения функциональной модели в формате IDEF0, моделей потоков данных в формате DFD, моделей базы данных в формате ERD разрабатываемой системы, т.е. необходимо разбиение системы на компоненты и описание взаимодействия этих компонентов.

IDEF0, зачастую называемая контекстной диаграммой, позволяет создать функциональную модель, структурно отображающую функции системы и среды, информацию и объекты, связывающие эти функции. Активно применяется также и для определения функций и требований разрабатываемой системы. Позволяет представить систему в виде набора взаимодействующих друг с другом подсистем с соответствующими входами/выходами, механизмами и управлениями, расположенными в иерархическом порядке для постепенного детализирования информации до определенного уровня.

Data Flow Diagramming (Диаграммы потоков данных) представляют функциональные требования к разрабатываемой системе в виде иерархии связанных потоками данных процессов, показывают каким образом происходит преобразование входных данных в выходные каждым процессом, выявляют отношения между рассматриваемыми процессами.

ER-диаграмма (ERD) является стандартной графической нотацией ER-модели (концептуальной модели данных предметной области), которая используется на концептуальном этапе проектирования баз данных и позволяет выделить ключевые сущности и связи между ними.

В настоящее время существует множество программных средств для построение вышеуказанных и множества других моделей, так как для решения данных задач сложился отдельный класс программных продуктов, примерами которых могут служить ARIS, Software Architect, BPWin и др.

В процессе выполнения данной работы как средство моделирования была выбрана кроссплатформенная система моделирования и анализа бизнес-процессов Ramus. Данного программное решение является удобным и понятным средством моделирования, частично совместимое с ERwin Process Modeler.

1.4. Средства разработки

В рамках данного проекта необходимо разработать динамический веб сайт, способный работать с реляционными базами данных.

Данный процесс можно условно разделить на три части – разработка видимой части сайта (т.е. пользовательского интерфейса), программно-административной части, а также самой базы данных.

Разработка пользовательского интерфейса сайта в основном ведется на языке разметки гипертекста – HTML (Hyper Text Markup Language). Для наиболее удобного управления стилями страниц используется каскадная таблица стилей – CSS (Cascading Style Sheets). Огромный вклад в интерактивность вносит один из сценарных языков программирования – Java Script.

Среди большого разнообразия языков для разработки программно-административной части программы наиболее целесообразным является выбор скриптового языка PHP (Hypertext Preprocessor).

Для разработки и поддержания баз данных было выбрано приложение phpMyAdmin. Данное приложение является бесплатным онлайн-сервисом для администрирования, написано на языке PHP и запускается непосредственно на серере.

1.4.1. HTML. CSS

HTML (Hyper Text Markup Language) – язык разметки гипертекста, предназначенный для написания гипертекстовых документов, публикуемых во всемирной паутина (WWW –World Wide Web). Является первым языком веб-программирования. Применяется в среде разработки веб-сайтов, являясь удобным, простым и понятным инструментом для верстки – процесса написания кода веб-страниц.

Командами в языке HTML являются теги, определяющие элементы html страницы, т.е. то, что при загрузке страницы представится пользователям. Такая страница представляет собой текстовый файл с указанием в нужной последовательности и необходимой степени вложенности друг в друга тегами. Порядок тегов и их синтаксис являются чрезвычайно важными при написании веб-страниц. Теги можно разделить на парные и одиночные.

Первые теги используются парами и заключены между двумя символами угловых скобок: <начальный тег> и </ конечный тег>. Между начальным и конечным тегом указывается содержимое элемента. Примеров парных тегов множество: <html> </html>, <table></table>, <p> </p> и другие.

Теги, не имеющие закрывающих (конечных) тегов, называют одиночными (непарными). Примерами таких тегов могут служить <br>, <hr> и многие другие. Обычно такие теги рекомендуется писать аналогично </br>.

Регистр символов для отображения тегов не важен, однако консорциум WWW (W3C), который занимается стандартизацией спецификации HTML, рекомендует использовать теги в нижнем регистре.

Простейший HTML-документ состоит из обязательных тегов, расположенных в строго определенном порядке:

* html - определяет начало и конец HTML-страницы.
* head - определяет головную часть HTML-страницы. Содержимое данного тега не отображается непосредственно в браузере (за исключением вложенного тега title), а призвано помочь браузеру в работе с данными.
* title - содержит название HTML-страницы.
* body – тело документа, т.е. отображаемая в окне браузера информация.

В настоящее время существует огромное количество HTML-тегов, помогающих реализовать самые разнообразные идеи. В Интернете можно найти большое количество html-справочников и таблиц основным HTML-тегов.

Для наиболее удобного управления внешним видом элементов существуют каскадные таблицы стилей – CSS. Данная технология позволяет задать параметры внешнего вида необходимых элементов. Так, например, в случае если необходимо выдержать несколько абзацев текста в одном стиле, совершенно не обязательно прописывать одно и тоже для каждого абзаца, а достаточно привести необходимые параметры в CSS-файле (или даже непосредственно в том же HTML-документе), указав при этом, к каким элементам применяются указанные параметры. Для точного указания элементов, к которым применяется тот или иной стиль, используются так называемые селекторы – части CSS-правила, указывающие браузеру конкретный элемент или группу элементов по определенному признаку – тег, класс, идентификатор и т.п.

1.4.2. JavaScript

JavaScript является сценарным языком программирования, широко применяющимся в веб-разработке, разработанный специально для приложений, работающих в сети интернет. Данный язык вносит огромный вклад в создание интерактивных динамических сайтов, так как позволяет вносить изменения в стили и содержание веб-страниц, отслеживать действия пользователя и реагировать на них, получить доступ к случайной части исходного кода страницы и многое другое.

Инструкции этого языка выполняются на стороне клиента, т.е. в браузере. На данный момент ни один современный браузер не обходится без поддержки JavaScript.

JavaScript-код представляет собой написанный по определенным правилам текстовый файл с расширение .js. Программный код на данном языке можно включить непосредственно внутрь HTML-документа либо сослаться на файл, его содержащий, используя при этом тег <script>.

Данный язык программирования постоянно совершенствуется, имеет огромное количество библиотек для решения самых разнообразных задач, удобен и практичен в использовании, прост в понимании.

Одной из самых популярных и используемых библиотек является jQuery – фокусирующаяся на взаимодействии JavaScript и HTML, значительно упрощающая и ускоряющая разработку веб-приложений и делающая синтаксис кода еще более понятным и прозрачным. Данная библиотека является кроссбраузерной, т.е. будет работать на любом браузере почти со 100% вероятностью. Библиотека jQuery хорошо зарекомендовала себя, постоянно обновляется и дополняется, пользуется популярности у множества программистов, а также в признанных мировых компаниях, таких как Microsoft и Google.

1.4.3. PHP

PHP является скриптовым языком, широко применяемым при веб-разработке. В отличии от уже рассмотренного JavaScript, PHP работает не на стороне клиента (браузера), а на стороне сервера, являясь таким образом серверным языком. При загрузке обычной HTML-страницы ее код сразу интерпретируется браузером и представляется пользователю. В случае с включением PHP скрипта, код сначала обрабатывается и выполняется сервером, и только потом результат выполнения интерпретируется клиентской стороной по описанным процедурам и правилам.

Данный язык предоставляет огромные возможности для создания динамических страниц, т.е. страниц, содержимое которых изменяется при наличии определенных условий. PHP прекрасно взаимодействует с HTML-кодом страницы. Можно как включить PHP-код в HTML-код страницы, так и выводить HTML-разметку при помощи PHP-кода.

Использование PHP-кода позволяет создавать действительно мощные сайты с возможностью продумывания и реализации «поведения» сайта.

Огромную роль PHP играет и при взаимодействии сайта с базой данных, так как при создании динамических страниц данные удобнее хранить именно в них, а не как в случае со статичными страницами непосредственно в HTML-коде. В подавляющем большинстве случаев осуществляется работа именно с MySQL.

PHP активно используется при разработке форм обратной связи, панелей администратора, возможностей регистрации, авторизации и комментирования, поиска по сайту, при работе с файлами и папками и во многих других случаях.

Данный язык программирования завоевал огромную популярность благодаря своей простоте, функциональности, скорости выполнения и расширяемости. Огромным плюсом является свободное распространение исходных кодов.

1.4.4. PHPMyAdmin

PHPMyAdmin является веб-приложением представляющее собой интерфейс для администрирования системой управления базами данных MySQL.

База данных - комплекс данных, структурированных и взаимосвязанных между собой. В свою очередь СУБД представляет собой комплекс языковых и программных средств, обеспечивающих управление созданием и использованием баз данных.

СУБД MySQL является одной из самых распространенных в мире, способна работать с такими языками программирования как PHP, Java, Perl и другими. Доступ к данным СУБД осуществляется посредством специального языка – SQL.

Подавляющим большинством сайтов активно и успешно используется связка PHP и MySQL.

На данный момент PHPMyAdmin является удобным и понятным средством работы с отдельными базами данных и таблицами или даже целыми серверами MySQL. Само приложение написано на любимом многими веб-разработчиками скриптовом языке PHP. В качестве языка работы с базами данных был выбран популярный и хорошо зарекомендовавший себя SQL.

Данное приложение имеет понятный и простой интерфейс, позволяет выполнять простейшие запросы к базам данных без непосредственного написания кода запроса. Также эта программа включается в пакет серверного софта большинства хостеров, является частью пакета программных локальных серверов, входит в состав популярных пакетов серверных программ, таких как Denwer и Apache.

ГЛАВА 2. Разработка комплекса моделей информационной системы

Подсистема электронной картотеки цифровых музыкальных композиций предназначена для удобного взаимодействия пользователей и администраторов с каталогом цифровых музыкальных медиафайлов (медиатекой) и базой метаданных музыкальных композиций.

Веб-приложение музыкальной картотеки предназначено для решения следующих задач:

* Работа с учётными данными пользователей: регистрация, авторизация, сохранение и редактирование пользовательских данных.
* Хранение и изменение метаданных музыкальных композиций.
* Хранение и изменение данных, хранимых в медиатеке.

Выполнение поисковых запросов пользователей – поиск данных в базе метаданных по разнообразным критериям.

Предоставление возможности воспроизведения медиафайлов, хранимых в медиатеке.

В рамках курсового проекта был рассмотрен процесс работы разрабатываемой системы, в соответствии с функциями которого были составлены:

1) функциональная модель (IDEF0);

2) модель потоков данных (DFD);

3) модель реляционной базы данных (ERD).

Функциональная модель разрабатывается с точки зрения администратора информационной системы.

Целью моделирования является наглядное представление взаимодействия блоков разрабатываемой системы.

2.1. Функциональная модель (IDEF0)

Целью методики IDEF0 является построение функциональной схемы исследуемой системы, описывающей все необходимые процессы с точностью, достаточной для однозначного моделирования деятельности системы [14].

Словарь функциональной модели разрабатываемой системы состоит из списка функциональных блоков (см. таблица 2.1.1), а также списка материальных и информационных потоков (см. таблица 2.1.2).

Внешними входными информационными потоками для модели являются:

* Метаданные о музыкальных композициях.
* Медиафайлы.
* Запрос пользователя.

Внешними выходными информационными потоками для модели являются:

* Найденные композиции.
* Прослушанная композиция.
* Выполненные запросы пользователя.

Внешними управляющими потоками для модели являются:

* Авторские права.
* Доверенные ресурсы.
* Правила хранения и организации данных в медиатеке.
* Правила хранения и организации данных в базе метаданных.
* Правила хранения и организации данных в базе пользовательских данных.

Основными механизмам для модели являются:

* Пользователь.
* Администратор.
* Веб-приложение.

На рисунках 2.1.1-2.1.5 представлены отдельные диаграммы функциональной модели.

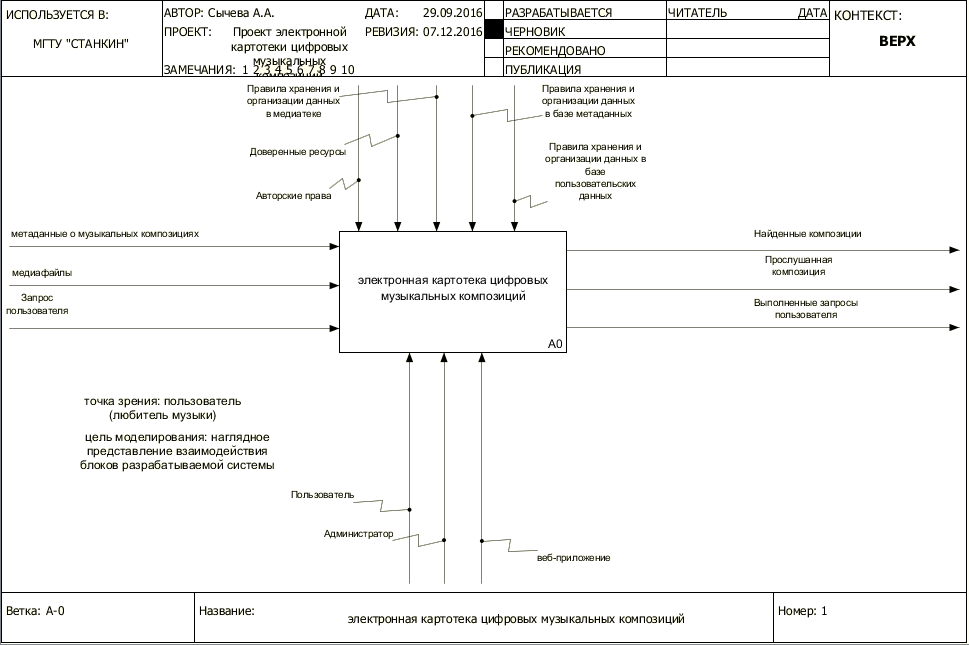


Рис. 2.1.1. Контекстная диаграмма.

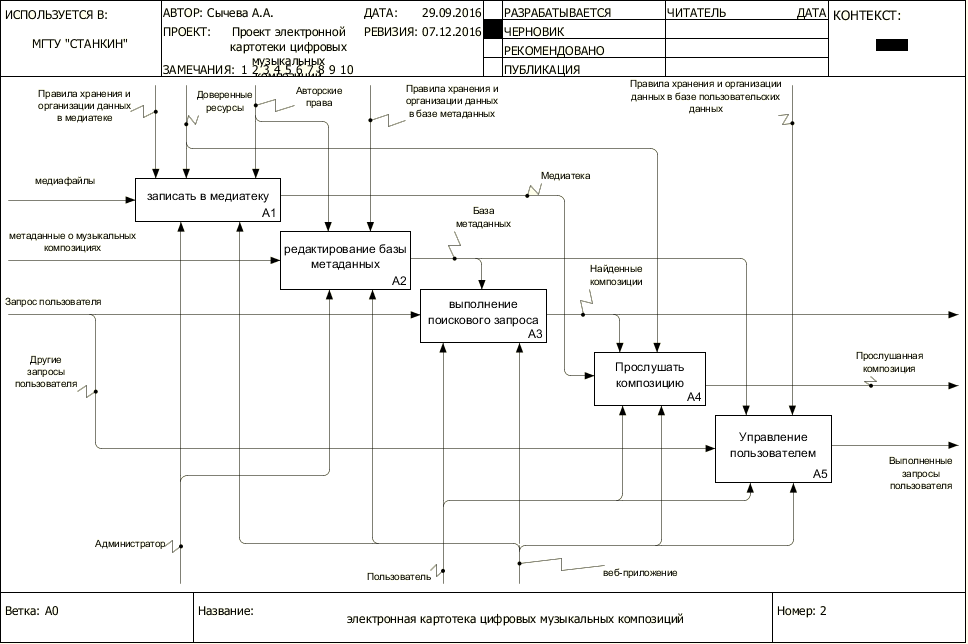


Рис. 2.1.2. Диаграмма процессов электронной картотеки цифровых музыкальных композиций.

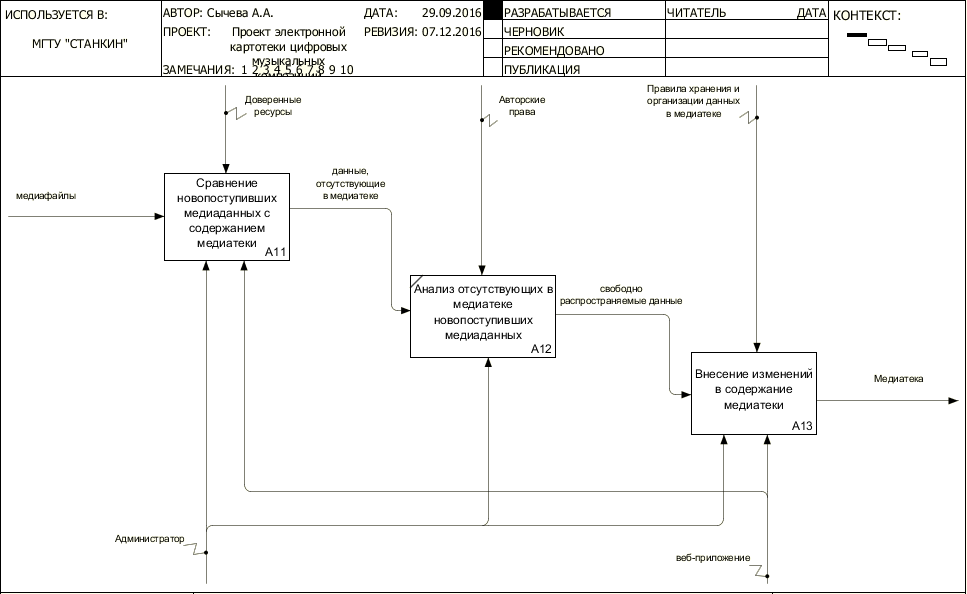


Рис. 2.1.3. Диаграмма процессов блока «Записать в медиатеку».

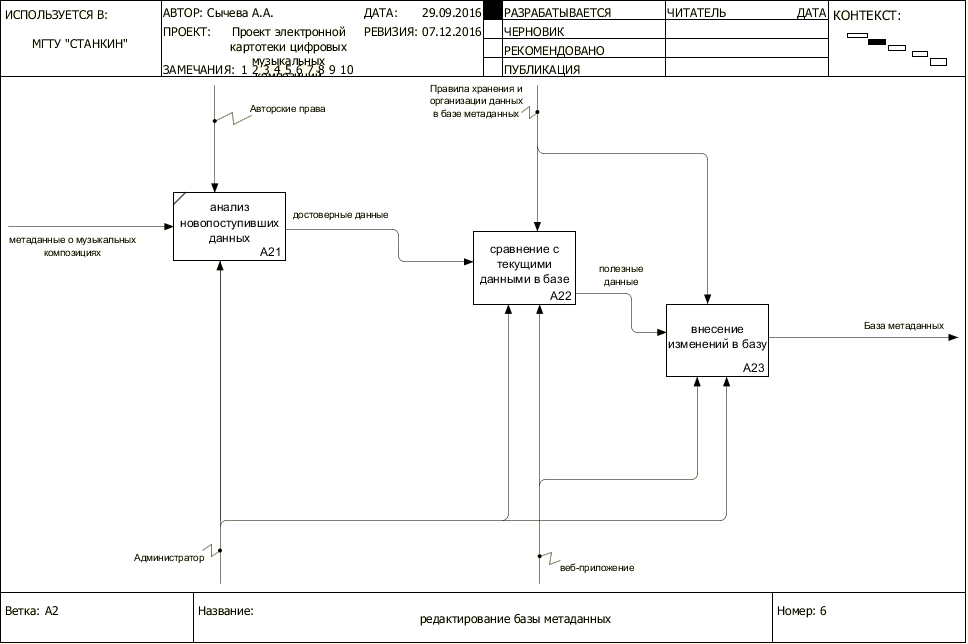


Рис. 2.1.4. Диаграмма процессов блока «Редактирование базы метаданных».

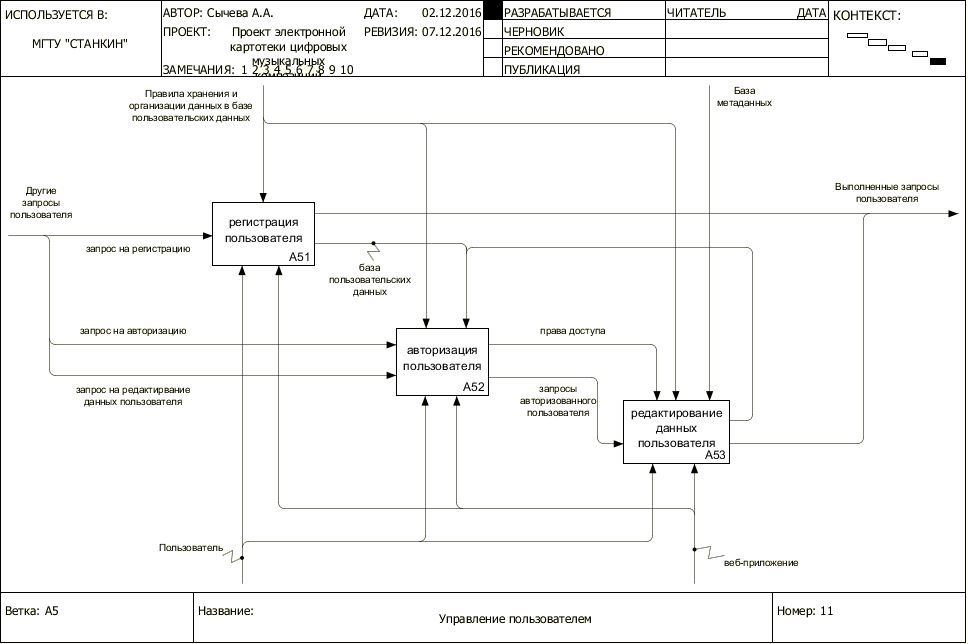
Рис. 2.1.5. Диаграмма процессов блока «Управление пользователем».

Таблица 2.1.1.

Список функциональных блоков.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Наименование | Определение | Форм | Данных | FP |
| A0 | Электронная картотека цифровых музыкальных композиций | Предназначена для удобного взаимодействия пользователей и администраторов с каталогом цифровых музыкальных медиафайлов и базой метаданных музыкальных композиций |  |  |  |
| A1 | Записать в медиатеку | Запись в медиатеку новопоступивших медиа данных | 3 | 1 | 9 |
| A2 | Редактирование базы метаданных | Внесение изменений в базу метаданных музыкальных композиций | 3 | 1 | 19 |
| A3 | Выполнение поискового запроса | Выполнение поступившего от пользователя запроса на поиск данных в базе метаданных |  |  |  |
| A4 | Прослушать композицию | Предоставление пользователю возможности воспроизведения медиа данных из медиатеки |  |  |  |
| A5 | Управление пользователем | Выполнение запросов на авторизацию и регистрацию пользователя, редактирование пользовательских данных | 3 | 1 | 19 |
| A11 | Сравнение новопоступивших медиданных с содержанием медиатеки | Проверка на дублирование данных медиатеки |  |  |  |
| A12 | Анализ отсутствующих в медиатеке новопоступивших медиаданных | Проверка возможности свободного распространения медиаданных |  |  |  |
| A13 | Внесение изменений в содержание медиатеки | Запись в медиатеку нового медиафайла или замена существующего |  |  |  |
| A21 | Анализ новопоступивших данных | Проверка достоверности новых метаданных |  |  |  |
| A22 | Сравнение с текущими данными в базе | Проверка на дублирование данных в базе метаданных |  |  |  |
| A23 | Внесение изменений в базу | Запись в базу метаданных новых данных или изменение существующих |  |  |  |
| A51 | Регистрация пользователя | Создание нового пользователя в системе |  |  |  |
| A52 | Авторизация пользователя | Выполнение входа пользователя в личный кабинет |  |  |  |
| A53 | Редактирование данных пользователя | Выполнение запроса пользователя на изменение своих данных |  |  |  |

Таблица 2.1.2.

Список материальных и информационных потоков.

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Определение |
| Администратор | Mechanism | Лицо, которое помимо возможностей пользователя имеет также возможность редактирования медиатеки и базы данных |
| Веб-приложение | Mechanism | Программная часть системы, упрощающая взаимодействие с данными |
| Выполненные запросы пользователя | Output | Успешно выполненные пользовательские запросы |
| Запрос пользователя | Input | Обращение пользователя к системе |
| Медиафайлы | Input | Аудио или видеоданные, которые предположительно возможно добавить в медиатеку |
| Метаданные о музыкальных композициях | Input | Данные о названии и авторе композиции, альбоме ее содержащего, годе выпуска и жанре. |
| Найденные композиции | Output | Результат поиска данных в базе метаданных |
| Пользователь | Mechanism | Лицо, которое имеет возможность просмотра данных медиатеки и базы метаданных, а также редактирования своих данных. |
| Прослушанная композиция | Output | Воспроизведенный пользователем медиафайл, содержащийся в медиатеке |

2.2. Модель потоков данных (DFD)

Диаграммы потоков данных (Data Flow Diagrams — DFD) представляют собой иерархию функциональных процессов, связанных потоками данных. Цель такого представления — продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами [15].

В некоторой степени каждый блок данных диаграмм можно принимать как отдельную форму в разрабатываемом программном продукте. Наименования таблиц базы данных информационной системы приводятся в формате «БД.Таблица». Медиатека же является отдельным хранилищем данных, а именно медиаданных, которые хранятся не в самой базе данных (базе метаданных), а на сервере в отельном каталоге.

На рисунках 2.2.1-2.2.9 представлены диаграммы потоков данных функциональных блоков системы. Не производилось построение диаграмм потоков данных для блоков, не взаимодействующих с информационной базой данных.

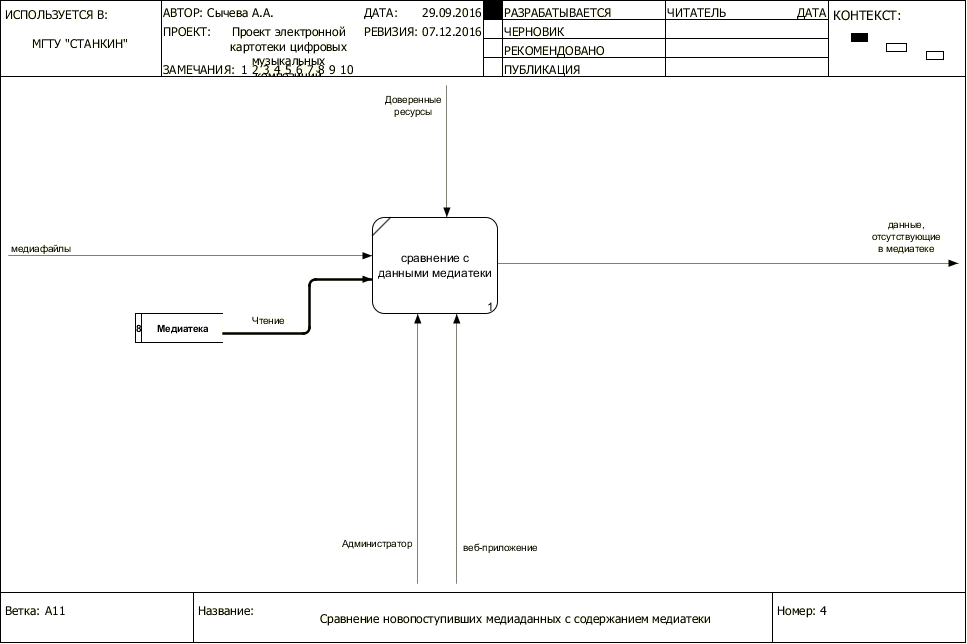


Рис. 2.2.1. Сравнение новопоступивших медиданных с содержанием медиатеки.

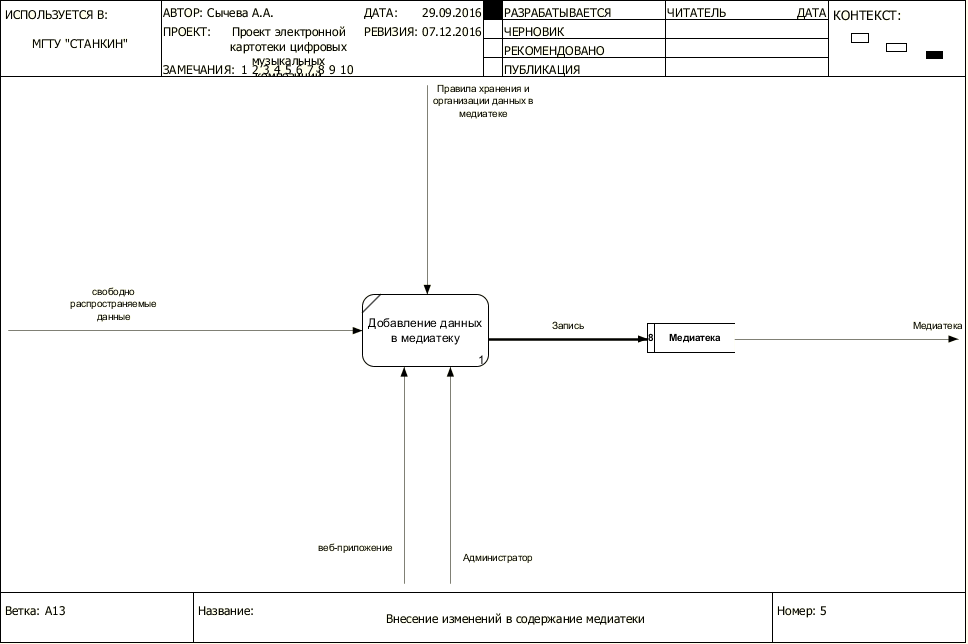


Рис. 2.2.2. Внесение изменений в содержание медиатеки.

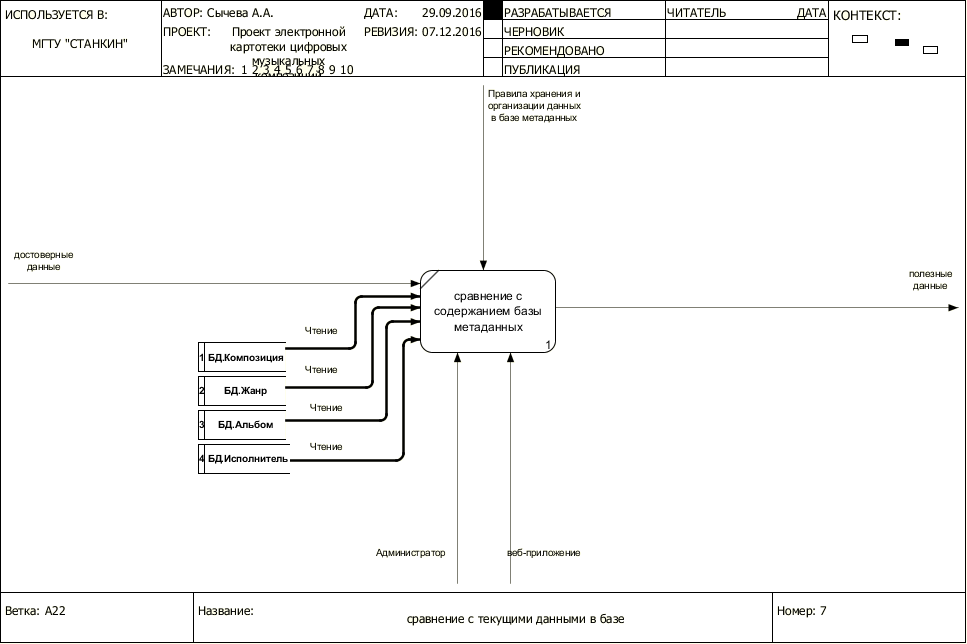


Рис. 2.2.3. Сравнение с текущими данными в базе.

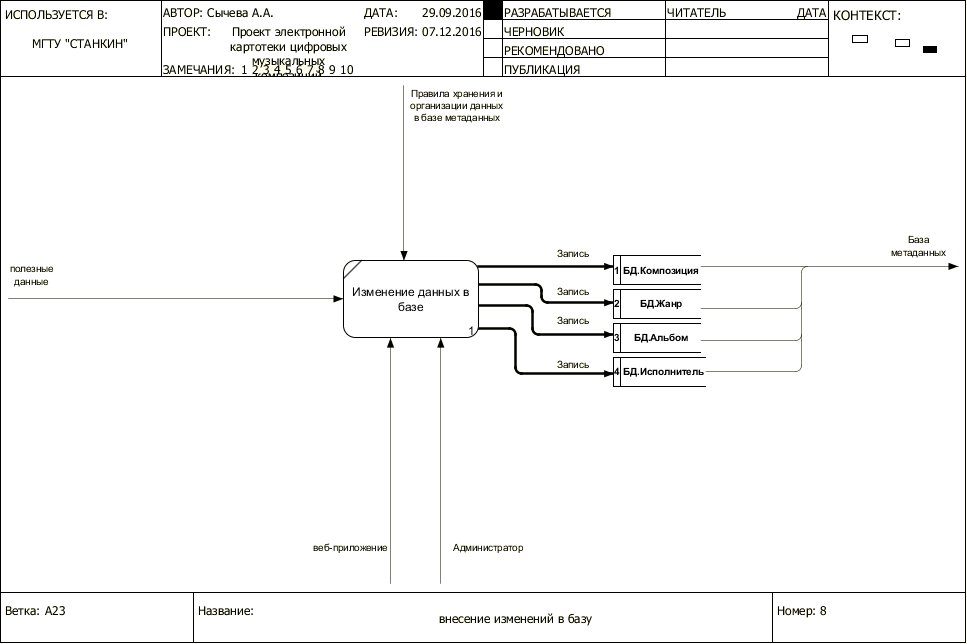


Рис.2.2.4. Внесение изменений в базу.

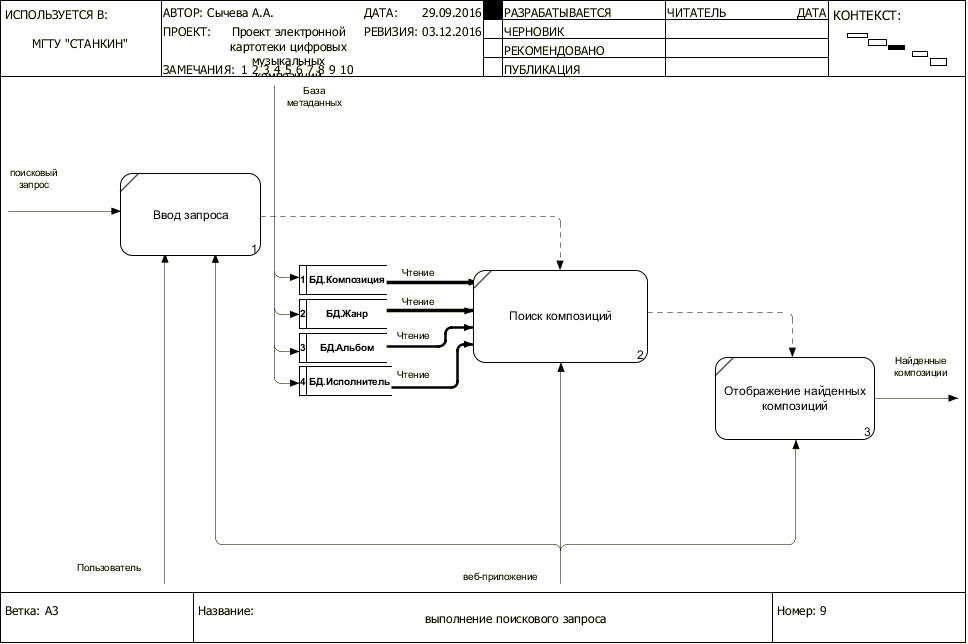


Рис. 2.2.5. Выполнение поискового запроса.

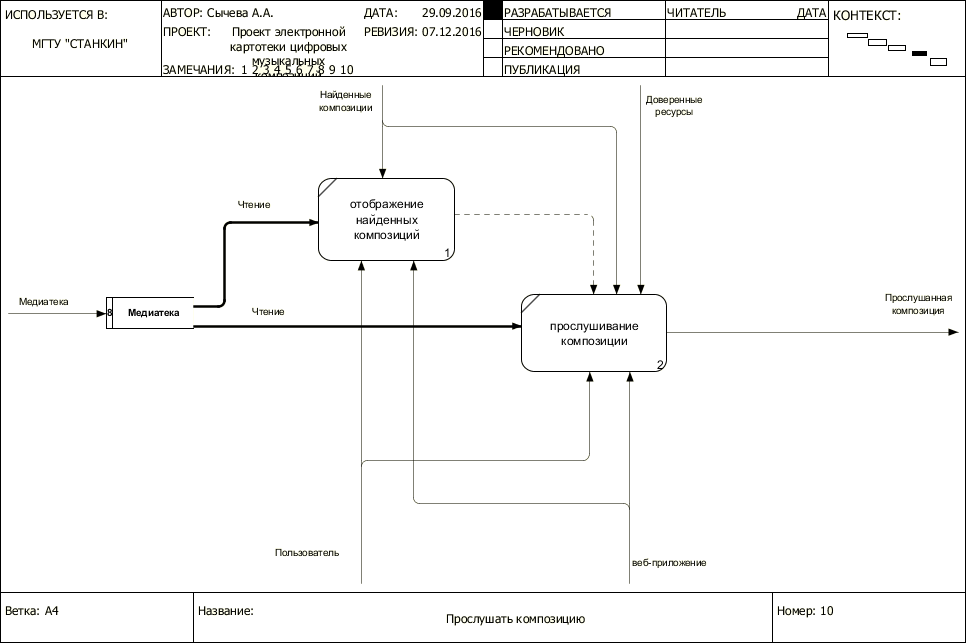


Рис. 2.2.6. Прослушать композицию.

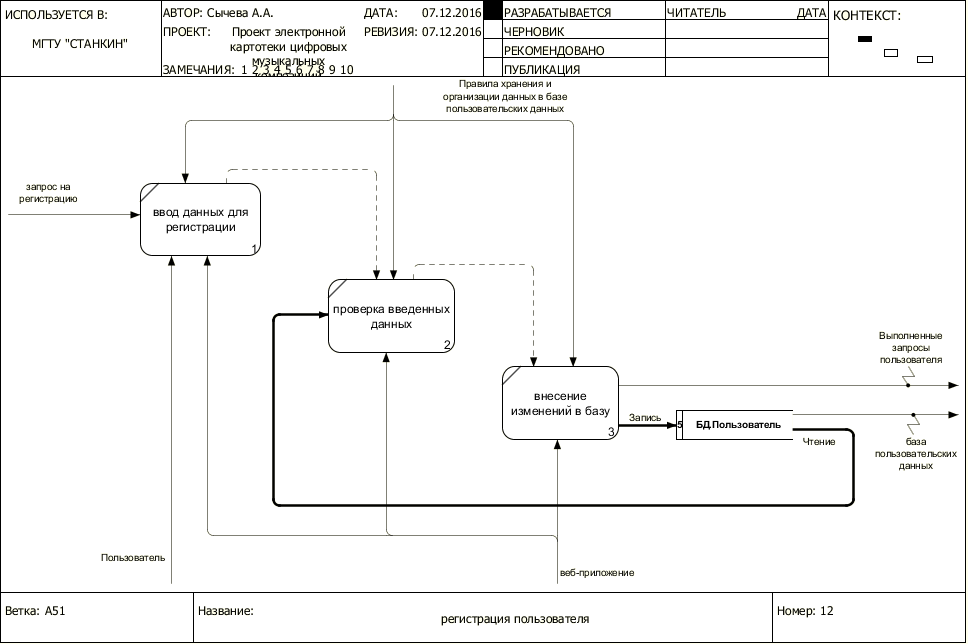


Рис. 2.2.7. Регистрация пользователя.

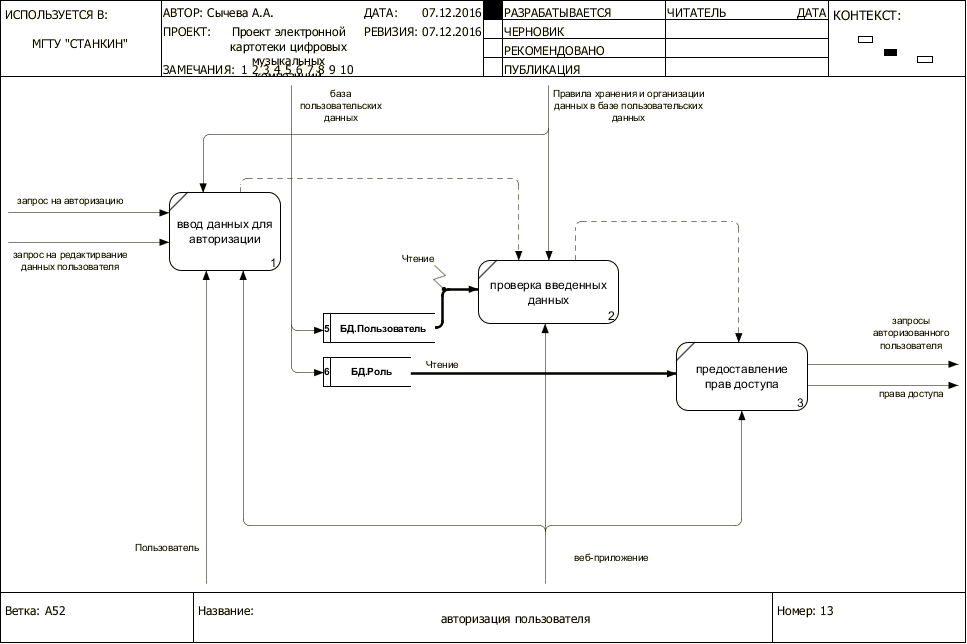


Рис. 2.2.8. Авторизация пользователя.

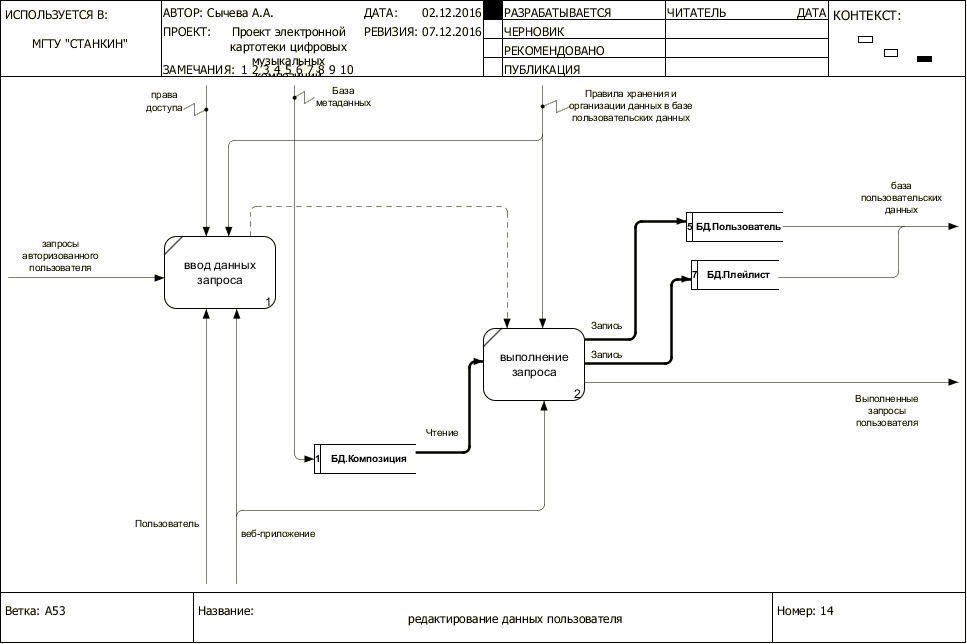


Рис. 2.2.9. Редактирование данных пользователя.

2.3. Реляционная модель базы данных (ERD)

ER-диаграмма является стандартной графической нотацией ER-модели (формальной конструкции для описания предметной области) [16]. Модель отображает структуру реляционной базы данных, т.е. таблицы, связи между ними и список атрибутов каждой таблицы.

Чтобы система была способна выполнить поставленные задачи, необходимо хранить данные о двух объектах: пользователь и композиция. В таком случае в информационной базе данных должны быть отражены следующие сущности:

* Пользователь – содержит данные пользователей: логин, пароль и роль (роль определяет права доступа к данным системы). Атрибуты: Логин пользователя, Пароль, Код роли.
* Роль – хранит возможные в системе роли. Атрибуты: Код роли, Название типа.
* Композиция – содержит в себе часть данных о музыкальной композиции, а именно название и наличие медиаданных, соответствующих данной записи в базе метаданных. Атрибуты: Код композиции, Название композиции, Медиа.
* Альбом - содержит в себе часть метаданных о музыкальной композиции, а именно альбом и год выхода альбома. Атрибуты: Код альбома, Название альбома, Год.
* Жанр - содержит в себе часть метаданных о музыкальной композиции, а именно название жанра. Атрибуты: Код жанра, Название жанра.
* Исполнитель - содержит в себе часть метаданных о музыкальной композиции, а именно наименование исполнителя. Атрибуты: Код исполнителя, Имя исполнителя.
* Плейлист – содержит соотношение пользователь-композиция для хранения пользовательских списков избранных композиций. Атрибуты: Логин пользователя, Код композиции.

После установления связей между данными сущностями из применения правил построения отношений дополнительно вытекает еще три сущности, необходимые для связи в реляционной модели базы данных.

Таким образом, на ER-диаграмме (рис. 2.3.1) представлено 7 сущностей:

* Композиция (Код композиции, Код исполнителя, Код альбома, Код жанра, Название композиции, Медиа)
* Альбом (Код альбома, Название альбома, Год)
* Жанр (Код жанра, Название жанра)
* Исполнитель (Код исполнителя, Имя исполнителя)
* Пользователь (Логин пользователя, пароль, Код роли)
* Роль (Код роли, Название роли)
* Плейлист (Логин пользователя, Код композиции)

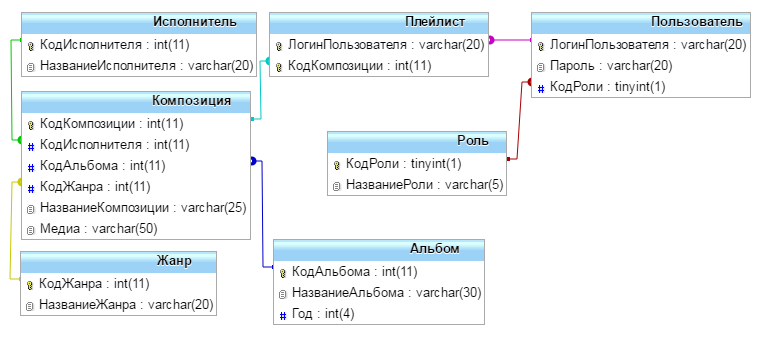


Рис. 2.3.1. Реляционная модель БД.

ГЛАВА 3. Разработка и исследовательские испытания прототипа программного и информационного обеспечения

3.1. Разработка прототипа

В процессе выполнения данной выпускной квалификационной работы была выполнена разработка (с построением моделей информационной системы в разделе 2 данного документа) и реализации при помощи указанных в разделе 1.4 средств разработки динамического сайта музыкальной картотеки.

Для ускорения процесса разработки и наискорейшего получения рабочего прототипа данной информационной системы все ее функции реализовывались в максимально упрощенном варианте. Основная функциональная нагрузка ориентирована на сторону клиента – браузер. Обращение к серверу происходит только при необходимости, т.е. невозможности выполнения действия при помощи только клиентской стороны.

Разработанный веб-ресурс действует в рамках одной страницы с расширением php. Таким образом, данной приложения относится к SPA – Single Page Application – приложение на одной странице. В процессе верстки применялся язык разметки гипертекста HTML и технологии блочной верстки сайтов, а также таблица каскадных стилей. Код данной страницы сайта и используемых стилей css представлен в приложении А.

Выполнение функций на клиентской стороне обеспечивается благодаря сценарному языку программирования JavaScript. Активно использовался фреймворк Bootstrap - свободный набор инструментов для создания сайтов и веб-приложений. Основные важные фрагменты JavaScript-кода представлены в приложении Б.

Как уже было сказано, для взаимодействия с серверной частью системы применялся скриптовый серверный язык PHP. Основными функциями, реализованными при помощи данного языка, являются работа с базой метаданных и медиатекой, а также вывод на страницу динамического контента, т.е. содержимого, которое нельзя предусмотреть и прописать заранее. Таким образом, данный язык активно используется в процессе регистрации, авторизации, отображения данных из базы мета данных и их изменении, при работе с базой метаданных, расположенной на сервере. Наиболее важные фрагменты кода на данном языке представлены в приложении В.

Главная страница для неавторизованного пользователя представлена на рисунке 3.1.1. Данная страница приветствует пользователя и дает понятие об основной функции данного ресурса. Также ставит пользователя в известность о том, что на данный момент ресурс работает в тестовом режиме, так как является прототипом.

Рис. 3.1.1. Главная страница сайта.

Для удобства работы с данным ресурсом реализованы две панели меню – горизонтальное меню навигации, расположенное сразу под шапкой сайта, и вертикальное меню разделов, изображенное слева страницы. Все содержимое сайта отображается в центральной части страницы посредством плавного скрытия и отображения контента соответственно нажатиям на кнопки панелей меню.

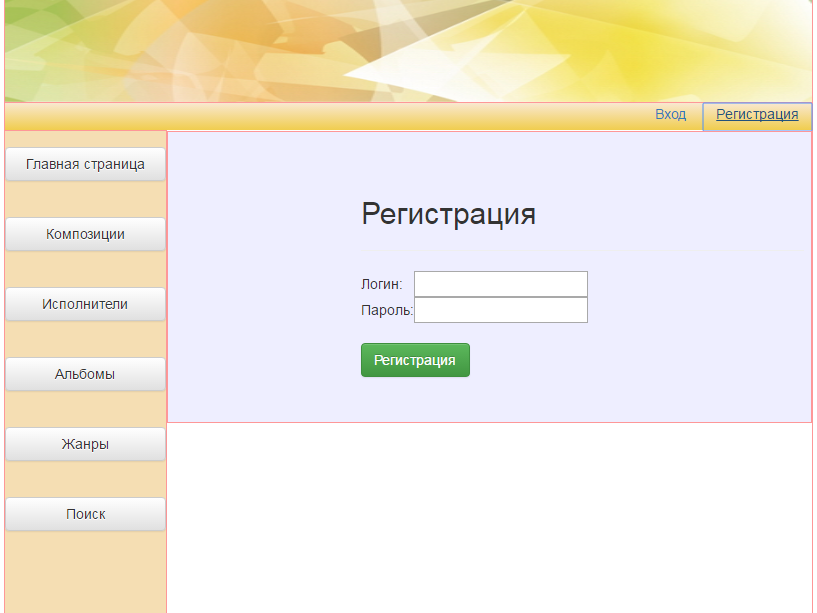
 Чтобы работать с данной системой необходимо авторизоваться. В случае отсутствия у пользователя аккаунта в системе необходимо пройти процесс регистрации. Данные функции запускаются нажатием на соответствующие кнопки панели навигации. Формы регистрации и авторизации представлены на рисунках 3.1.2 и 3.1.3.

Рис. 3.1.2. Форма регистрации.

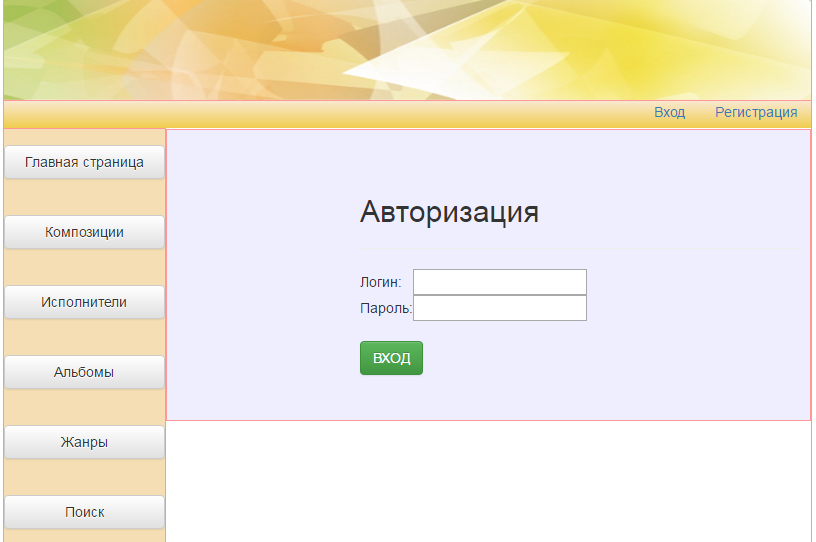
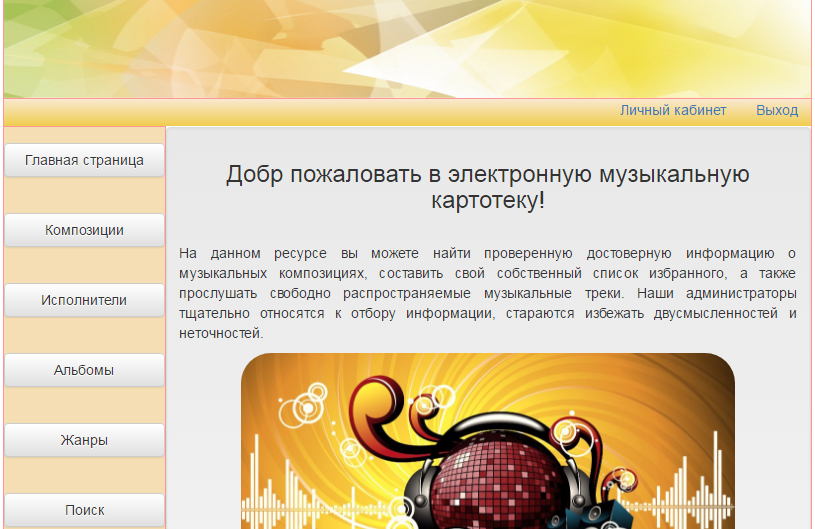
Система интерактивно реагирует на действия пользователя – в случае неверного указания необходимых данных выводятся соответствующие сообщения об ошибках. После успешного процесса авторизации предоставляется доступ во все разделы меню разделов. Панель навигации также соответственно отображает кнопки для входа в личный кабинет, выхода из системы и администрирования системы (в случае авторизации с аккаунта администратора), что представлено на рисунках 3.1.4 и 3.1.5. В дальнейшем система будет рассматриваться с аккаунта администратора.

Рис. 3.1.3. Форма авторизации.

Кнопка выхода используется для завершения (обнуления) текущей сессии и выхода из аккаунта. После этого пользователь вновь перенаправляется на главную страницу сайта, представленную на рисунке 3.1.1.

Авторизовавшийся пользователь имеет возможность просмотреть содержимое базы метаданных музыкальных композиций по разделам, указанным в меню разделов – Композиции, Исполнители, Альбомы, Жанры. Примеры отображения данных из разделов «Композиции» и «Исполнители» представлены на рисунках 3.1.6 и 3.1.7. Отображение данных в остальных разделах происходит совершенно аналогично.

Рис. 3.1.4. Видение сайта авторизовавшимся пользователем.

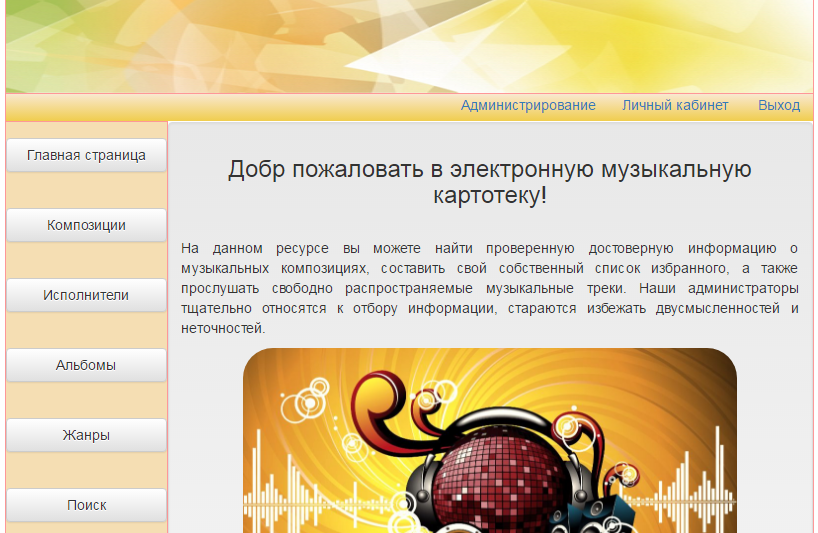


Рис. 3.1.5. Видение сайта авторизовавшимся администратором.

.

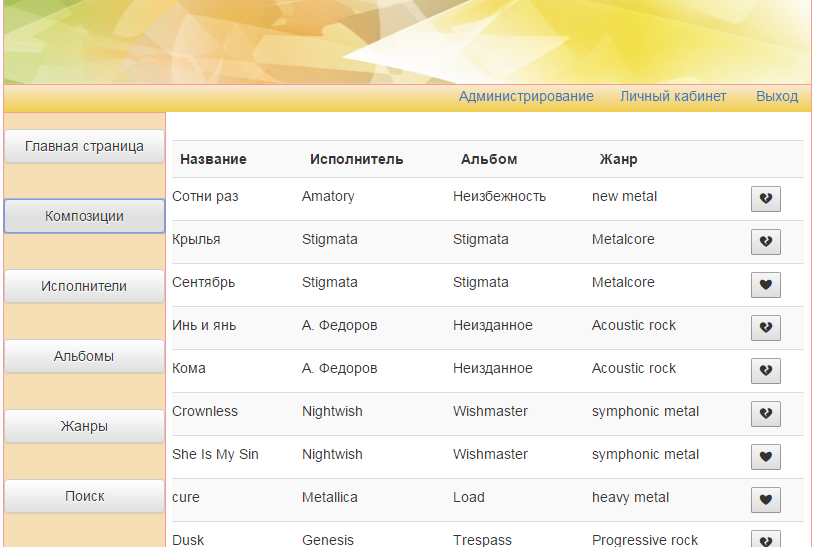
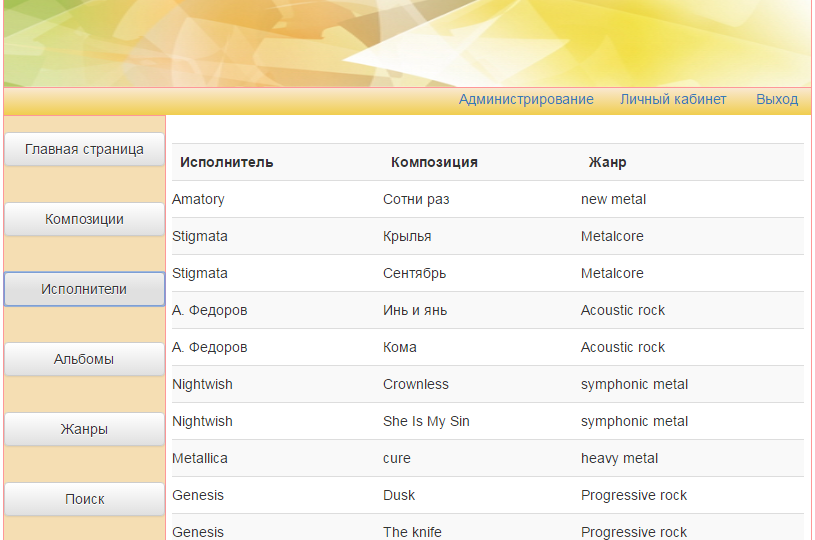
Рис. 3.1.6. Раздел композиций. 

Рис. 3.1.7. Раздел исполнителей.

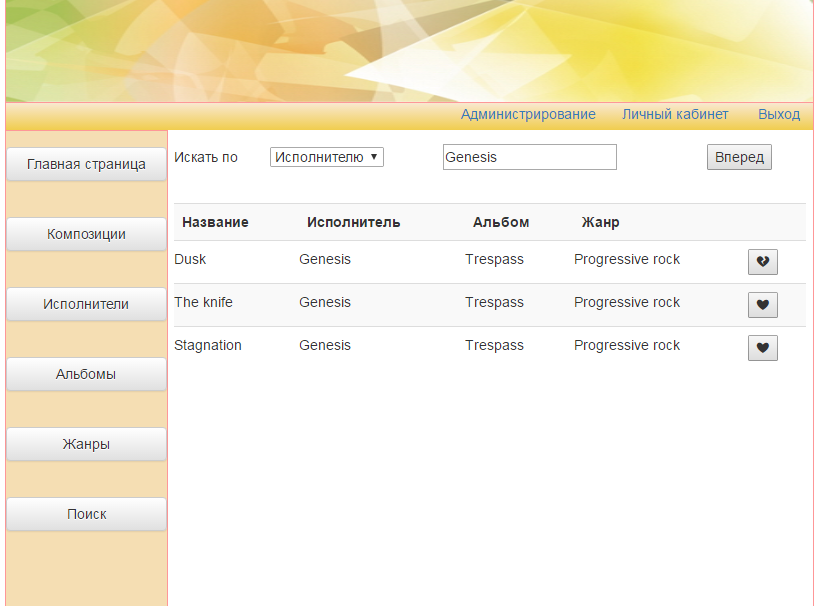
Меню разделов также имеет пункт «Поиск», который предоставляет пользователю возможность поиска данных в системе по выбранному им разделу: композиция, исполнитель, жанр, альбом. Пример работы поиска представлен на рисунке 3.1.8.

Рис. 3.1.8. Поиск по исполнителям.

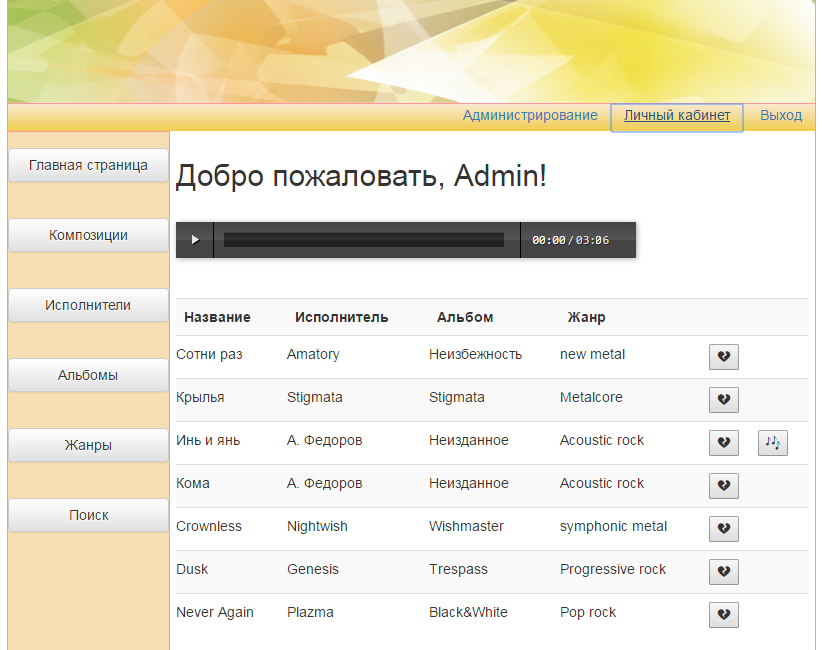
Свой список избранного пользователь может просмотреть и отредактировать в своем личном кабинете, открыв его по нажатию соответствующей кнопки на панели навигации. Кнопка удаления из плейлиста расположена и выглядит также, как при просмотре раздела композиций или выполнении поиска. При просмотре плейлиста из личного кабинета пользователю также предоставляется возможность прослушать понравившуюся композицию, в случае если медиафайл данной композиции имеется в системе, через встроенный в страницу плеер. Для прослушивания композиции необходимо нажать на кнопку с изображением нот, расположенной в последнем столбце таблицы, отображающей плейлист пользователя. Личный кабинет представлен на рисунке 3.1.9.

Рис. 3.1.9. Личный кабинет.

Помимо вышеперечисленных функций, доступных рядовому пользователю, администратору предоставляется возможность внесения изменений в базу метаданных и медиатеку. Для этого необходимо перейти в раздел администрирования посредством нажатия соответствующей кнопки панели навигации. Страница администрирования системы представлена на рисунке 3.1.10.

Под внесением изменений в базу метаданных подразумевается добавления новых данных (в том числе добавление новой композиции и связывание ее с уже существующими исполнителями, альбомами и жанрами), удаление или переименование уже существующих данных. Так как система подразумевает хранение полной и достоверной информации о музыкальных композициях, возможность частичного добавления данных отсутствует. Например, в случае добавления композиции без исполнителя или альбома.

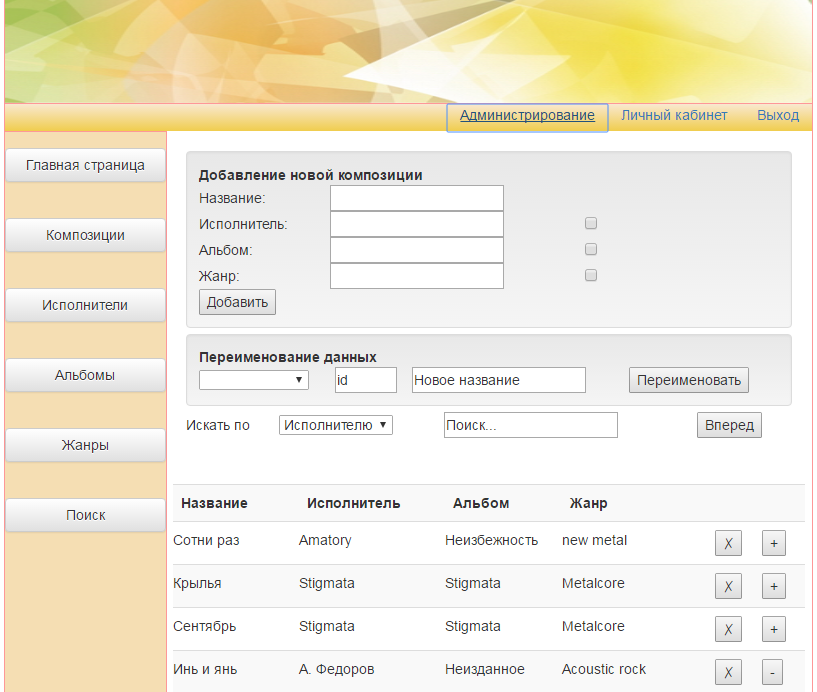


Рис. 3.1.10. Администрирование.

Под внесением изменений в медиатеку подразумевается добавление медиадынных на сервер или их удаление оттуда. Параллельно этому происходят также некоторые изменения и в метаданных композиций – необходимо отобразить удаление или добавление медиаданных конкретной композиции.

Процесс добавления или переименования метаданных выполняется в соответствующих разделах страницы администрирования, расположенных перед поиском по таблице, отображающей все композиции (аналогично пункту композиций меню разделов.) .

Для добавления абсолютно новых данных в полях для исполнителя, альбома и жанра указываются названия, ставятся галочки в чекбоксах напротив данных полей. В таком случае также появится поле для указания года альбома. После необходимо нажать соответствующую кнопку.

Для связывания новой композиции с уже существующими данными, необходимо снять галочки у полей уже существующих данных и указать вместо названия индивидуальные коды, которые можно быстро получить кликом по соответствующему полю таблицы внизу данной страницы. После необходимо нажать соответствующую кнопку.

При переименовании данных необходимо указать категорию редактируемых данных (исполнитель, композиция, альбом или жанр), а также индивидуальный код. Данные поля автоматически заполняются при клике на ячейку таблицы внизу страницы. Администратору остается только указать новое название и нажать соответствующую кнопку.

Удаление композиции, а также добавление и удаление медиаданных выполняется при помощи кнопок, указанных в последних двух столбцах таблицы внизу страницы.

Кнопка предпоследнего столбца таблицы (с изображением креста) выполняет полное удаление данной позиции, т.е. композиции (удаление метаданных) и связанных с ней медиафайлов из системы.

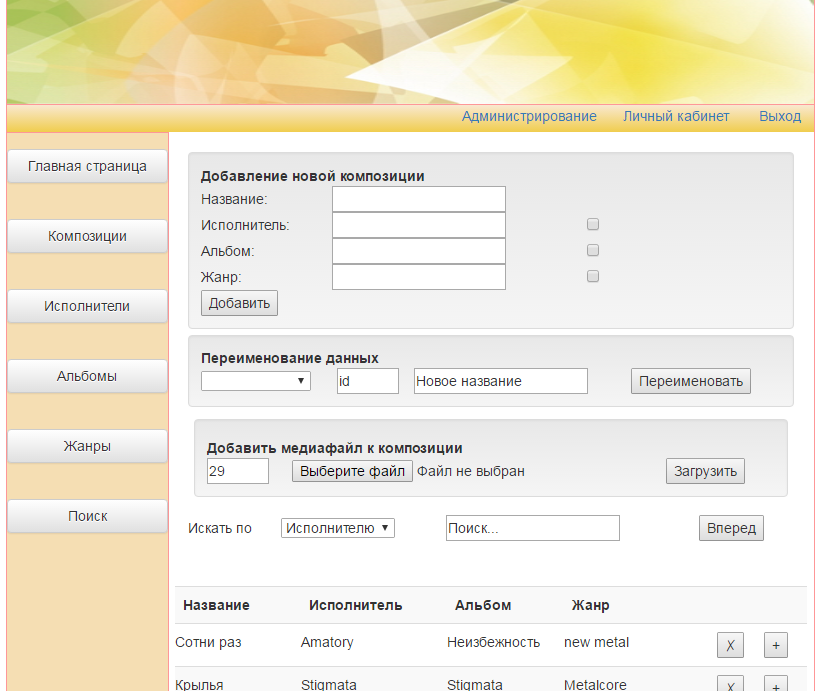
В последнем столбце располагается кнопка для удаления или добавления медиафайлов (в зависимости от их наличия в системе с изображением знака плюса или минуса). При нажатии на кнопку добавления медиафайлов после раздела о переименовании данных открывается раздел для загрузки файлов, который можно увидеть на рисунке 3.1.11. В данном разделе администратору необходимо лишь выбрать загружаемый файл и нажать кнопку загрузки. Поле с индивидуальным кодом при этом заполняется автоматически

Рис. 3.1.11. Администрирование. Добавление мадиафайла.

Как и в случае с авторизацией и регистрацией система реагирует на действия администратора, сообщая о возникающих в процесс администрирования ошибках. Подобные ошибки могут быть связанны с некорректным вводом администратором необходимых данных (пустые поля, ввод отрицательного числа в разделе года альбома и т.д.).

3.2. Испытания прототипа

После реализации задуманной системы необходимо провести испытания – т.е. протестировать систему, проверить на сколько она соответствует заявленным требованиям и на сколько хорошо выполняет свои функции. В данном контексте объектом испытаний служит разрабатываемый веб-сайт.

Главным требованием успешного прохождения испытаний при тестировании разрабатываемого в данной выпускной квалификационной работе сайта является выполнение системой ее функций, подробно описанных в главе 2.

Работа с учётными данными пользователей: регистрация, авторизация, сохранение и редактирование пользовательских данных – успешно реализована. Система адекватно реагирует на запросы пользователя на авторизацию, регистрацию и изменение пользовательских данных (изменение списка избранных композиций), сообщает об ошибках, вызванных некорректными действиями пользователя. Ошибок со стороны системы не обнаружено.

Хранение и изменение метаданных музыкальных композиций – успешно реализована. Система дает администратору возможность через простой и понятный интерфейс вносить изменения в базу метаданных – добавление, редактирование (переименование), удаление данных. Сообщает об ошибках, вызванных некорректными действиями администратора. Ошибок со стороны системы не обнаружено.

Хранение и изменение данных, хранимых в медиатеке –успешно реализована. Администратору предоставляется возможность быстро и без излишних сложностей добавлять или удалять медиаданные к композициям. Необходимые для реализации этой функции изменения метаданных композиций производятся в автоматизированном режиме. Система сообщает об ошибках, вызванных некорректными действиями администратора. Ошибок со стороны системы не обнаружено.

Успешность выполнения вышеуказанных функций можно наблюдать на скриншотах из предыдущего раздела.

Помимо успешной реализации задуманных функций также стоит протестировать и определенные характеристики разработанной системы.

Благодаря бесплатному сервису PageSpeed Insights удалось получить оценочные суждения о скорости загрузки страницы. Данный сервис дал разработанному ресурсу оценку в 75 из 100 баллов. Для улучшения данного показателя рекомендуется оптимизировать изображения, удалить блокирующий загрузку контента JavaScript и CSS код, сократить количество JavaScript кода, использовать кэш браузера.

Более широкое понятие о характеристиках разработанной системы дает сервис Load Impact, имитирующий в течении 5 минут обращение к выбранному для тестирования сайту 25 пользователей и представляющий результаты тестирования в виде наглядных графиков, иллюстрирующих изменение характеристик сайта в течении времени тестирования (см. рис. 3.2.1-3.2.6). Под VUs active на представленных графиках подразумевается количество активных виртуальных пользователей.

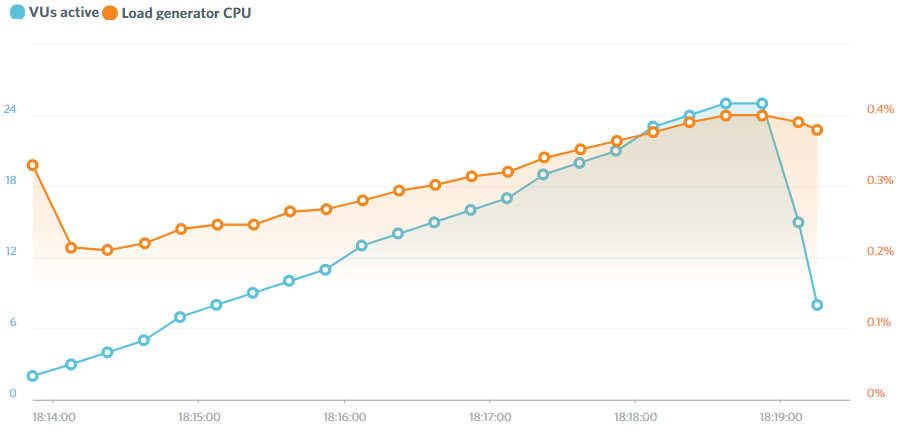
Из представленных графиков можно сделать вывод, что реализованный веб-сайт довольно успешно справляется с увеличением количества пользователей.

Рис. 30. Загруженность процессора сервера

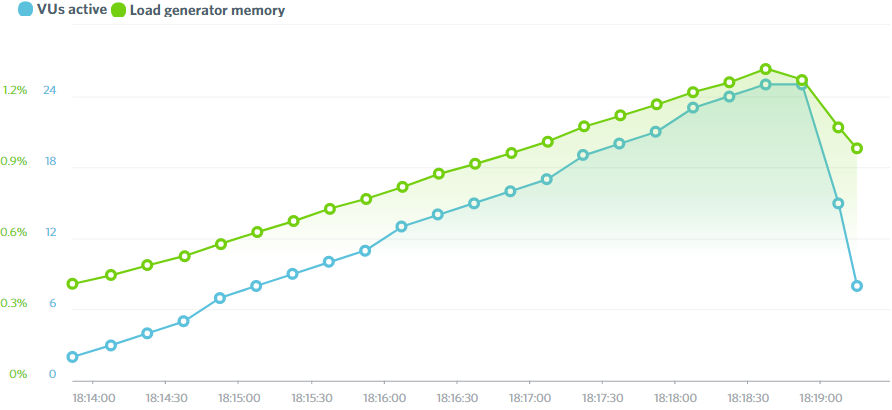


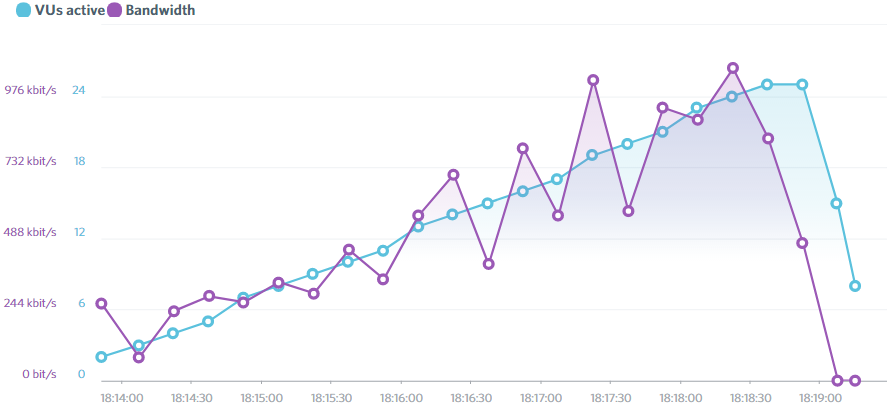
Рис. 31. Загруженность памяти сервера.

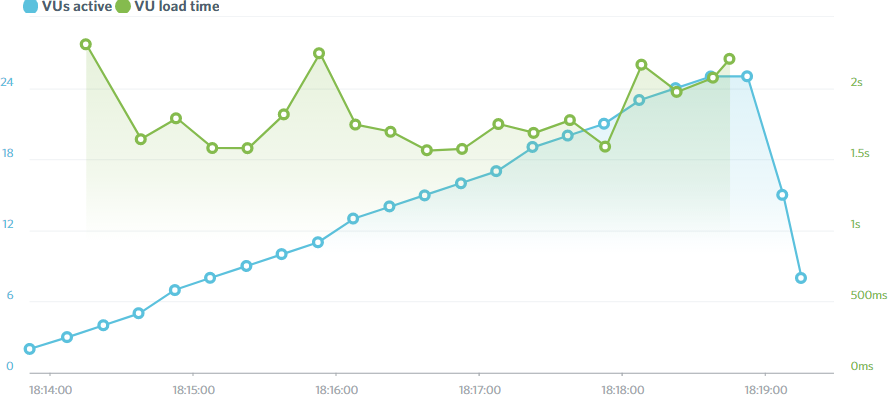
Рис. 32. Ширина полосы пропускания. 

Рис. 33. Время загрузки страницы.

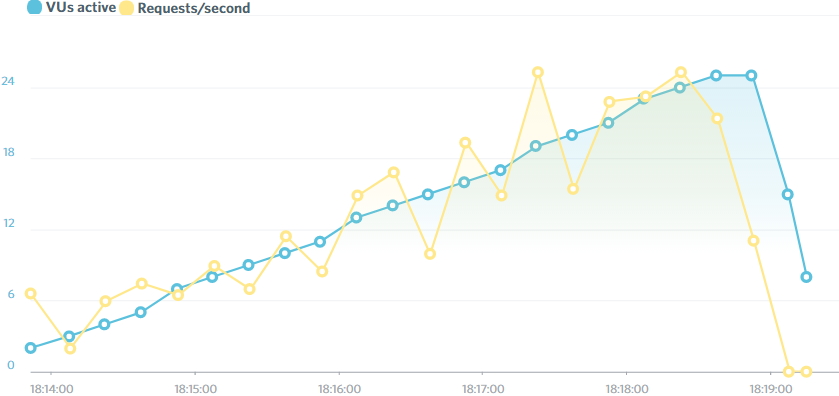


Рис. 34. Количество выполненных за секунду запросов.

Исходя из всего вышесказанного в данном разделе, можно сделать вывод, что разработанная система выполняет все запланированные для реализации функции в удобном и понятном для пользователя виде. В процессе тестирования не было установлено актов выхода системы из строя. Все свои функции система выполняет оперативно и практически незаметно для использующего ее лица. Для повышения производительности работы системы и снижения нагрузки на сервер активно применялись технологии работы на стороне клиента. Также в процессе тестирования были сформулированы рекомендации для ускорения процесса загрузки страницы.

ЗАКЛЮЧЕНИЕ

В настоящее время информационные процессы играют огромную роль в жизни общества. В любой сфере деятельности именно информация является стратегически важным ресурсом. Благодаря популярности и широкому распространению глобальной сети информационные процессы все чаще протекают в Интернете, а не настольных приложениях пользователей. Накопление впечатляющих объемов данных требует решения проблемы их хранения, которым и являются базы данных.

Поэтому именно в эту эпоху развития общества все актуальнее становится разработка веб-сайтов, взаимодействующих с базами данных. Подобные решение пользуются огромной популярностью у самых разнообразных категорий населения в различных областях.

Современные технологии предоставляют впечатляющие возможности для разработки различных веб-решений. Каждый программист может подобрать подходящие именно ему технологии для выполнения данного процесса.

Наиболее часто используемой и хорошо себе зарекомендовавшей себя технологией является совместное использование языка разметки гипертекста – HTML, каскадных таблиц стилей – CSS, скриптовых языков JavaScript и PHP и СУБД MySQL. Вместе данные технологии позволяют создать по-настоящему мощные, интерактивные и удобные веб-ресурсы, будь то просто сайт интернет-магазина или система моделирования для научных изысканий.

В данной работе проводилось построения моделей информационной системы музыкальной картотеки, а после ее реализации при помощи совокупного применения вышеуказанных технологий. Оценка трудоемкости и сроков создание разрабатываемой информационной системы проводился по двум методам - FPA IFPUG и COCOMO II.

Расчеты, выполненные первым методом, позволяют оценить сложность требуемых для создания информационной системы программных средств в 57 выровненных функциональных точек, а объем программного кода на языках программирования высокого уровня - в 2537 строк кода.

Расчеты, выполненные вторым методом, позволяют оценить общие трудозатраты проекта разработки программных средств в 11 человеко-месяцев, а ожидаемую продолжительность проекта – в 7 месяцев.

Разработанный в процессе выполнения выпускной квалификационной работы ресурс обладает удобным и понятным интерфейсом. хорошо показал себя во время проведения испытаний. Были сделаны выводы, касающиеся улучшения и оптимизирования работы системы.

Разработанный веб-сайт наглядно представляет успешность и удобство разработки и использования вышеуказанных технологий. Аналогичные разработки удобно будет производить также и в других предметах областях. В дальнейшей перспективе имеется возможность усовершенствования реализованного ресурса, под чем можно понимать как повышение производительности и оптимизация ресурса, так и расширение предметной области или перепрофилирование на другие сферы жизни общества.

Список источников

1. Сайт «Центр прикладных исследований компьютерных сетей» [Электронный ресурс] – Режим доступа http://arccn.ru/media/572/, свободный. Дата обращения 23.04.17
2. ГОСТ Р ИСО 9001-2015 Системы менеджмента качества. Требования.
3. ГОСТ 34.003-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения.
4. ГОСТ Р ИСО/МЭК 12207-2010 Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств.
5. ГОСТ Р 56923-2016/ISO/IEC TR 24748-3:2011 Информационные технологии. Системная и программная инженерия. Управление жизненным циклом. Часть 3. Руководство по применению ИСО/МЭК 12207 (Процессы жизненного цикла программных средств).
6. ГОСТ 24026-80 Исследовательские испытания. Планирование эксперимента. Термины и определения.
7. Методология IDEF0. Стандарт.
8. Сайт «НОУ Интуит. Проектирование информационных систем» [Электронный ресурс] – Режим доступа http://www.intuit.ru/studies/courses/2195/55/lecture/1632?page=2, свободный. Дата обращения: 02.05.17.
9. Росс В. С. Создание сайтов: HTML, CSS, PHP, MySQL. Учебное пособие, ч. 1 — МГДД(Ю)Т, М.:2010 – 107 с.
10. Сайт «ЯКласс» [Электронный ресурс] – Режим доступа http://www.yaklass.ru/materiali?chtid=511&mode=cht, свободный. Дата обращения 24.04.17.
11. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 3-е изд. — СПб.: Питер, 2015. — 688 с.
12. Сайт «Современный учебник JavaScript» [Электронный ресурс] – Режим доступа https://learn.javascript.ru/, свободный. Дата обращения 25.04.17.
13. Сайт «PHP-MyAdmin по-русски» [Электронный ресурс] – Режим доступа http://php-myadmin.ru/, свободный. Дата обращения 24.04.17.
14. Сайт «НОУ ИНТУИТ | Проектирование информационных систем» [Электронный ресурс] – Режим доступа: http://www.intuit.ru/studies/courses/2195/55/info, свободный. Дата обращения: 15.05.17.
15. Сайт «E-educ» [Электронный ресурс] – Режим доступа: http://e-educ.ru/bd14.html, свободный. Дата обращения: 15.05.17.
16. Сайт «Ронл» [Электронный ресурс] – Режим доступа: http://www.ronl.ru/lektsii/informatika/868901/, свободный. Дата обращения: 16.05.17.

ПРИЛОЖЕНИЕ А

Код основной страницы веб-приложения

<?

session\_start();

?>

<!DOCTYPE html>

<html>

<head>

<script src="https://code.jquery.com/jquery-3.1.0.js" integrity="sha256-slogkvB1K3VOkzAI8QITxV3VzpOnkeNVsKvtkYLMjfk=" crossorigin="anonymous"></script>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Блочная вёрстка</title>

<link rel="stylesheet" type="text/css" href="css/1.css">

<link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.3.2/css/bootstrap.min.css">

<link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.3.2/css/bootstrap-theme.min.css">

<script src="//netdna.bootstrapcdn.com/bootstrap/3.3.2/js/bootstrap.min.js"></script>

<script src="http://thebloodarrow.ru/audiojs/audio.min.js"></script>

</head>

<body>

<div id="header"></div>

<div id="navigation">

<div id="index\_navigation" class="pull-right" >

<input type='button' id='login' class="btn btn-link" value='Вход'/>

<input type='button' id='registration' class="btn btn-link" value='Регистрация'/>

</div>

<div id="user\_navigation" class="hide\_div pull-right">

<form action="includes/logout.php" method="POST">

<input type='button' id='profile' class="btn btn-link" value='Личный кабинет'/>

<input type='submit' id='logout\_button' class="btn btn-link" value='Выход'/>

</form>

</div>

<div id="admin\_navigation" class="hide\_div pull-right">

<input type='button' id='admin\_panel' class="btn btn-link" value='Администрирование'/>

</div>

</div>

<div id="menu">

<input type='button' id='main\_page' class="btn btn-default btn-block" value='Главная страница'/><br>

<input type='button' id='all\_comp' class="btn btn-default btn-block" value='Композиции'/><br>

<input type='button' id='all\_isp' class="btn btn-default btn-block" value='Исполнители'/><br>

<input type='button' id='all\_album' class="btn btn-default btn-block" value='Альбомы'/><br>

<input type='button' id='all\_genre' class="btn btn-default btn-block" value='Жанры'/><br>

<input type='button' id='search\_button' class="btn btn-default btn-block" value='Поиск'/><br>

</div>

<div id="content">

<div id="main\_page\_div">

<div class="well" id="main\_div">

<h3 align="center">Добр пожаловать в электронную музыкальную картотеку!</h3><br>

<p align="justify"> На данном ресурсе вы можете найти проверенную достоверную информацию о музыкальных композициях, составить свой собственный список избранного, а также прослушать свободно распространяемые музыкальные треки. Наши администраторы тщательно относятся к отбору информации, стараются избежать двусмысленностей и неточностей.</p>

<img src="http://www.thebloodarrow.ru/hKtssxX.jpeg" align="center"/>

<p align="justify">На данный момент ресурс работает в тестовом режиме и выполняет свои минимальные функции.</p>

</div>

</div>

<div id="login\_div" class="hide\_div">

<h2>Авторизация</h2>

<span id='errlogin'></span><hr>

<table>

<tr>

<td>Логин:</td>

<td> <input type='text' id='login\_name'/><br></td>

</tr>

<td>Пароль: </td>

<td><input type='password' id='login\_pass'><br></td>

</table><br>

<input type='button' id='login\_button' class="btn btn-success" value='ВХОД'/>

</div>

<div id="reg\_div" class="hide\_div">

<h2>Регистрация</h2>

<span id='errreg'></span><hr>

<table>

<tr>

<td>Логин: </td>

<td><input type='text' id='reg\_name'/><br></td>

</tr>

<tr>

<td>Пароль: </td>

<td><input type='password' id='reg\_pass'><br></td>

</tr>

</table><br>

<input type='button' id='reg\_button' class="btn btn-success" value='Регистрация'/>

</div>

<div class="hide\_div" id="personal\_page">

</div>

<div class="hide\_div" id="admin\_page">

<div class="well">

<table class="inv\_table">

<tr><th colspan="2">Добавление новой композиции</th><th></th></tr>

<tr>

<td>Название:</td>

<td><input type="text" id="add\_comp"></td>

</tr>

<tr>

<td>Исполнитель:</td>

<td><input type="text" id="add\_ispcomp"></td>

<td><input type="checkbox" name="2" id="ch\_isp"/></td>

</tr>

<tr>

<td>Альбом:</td>

<td><input type="text" id="add\_albcomp"></td>

<td><input type="checkbox" name="3" id="ch\_alb" onclick="ch\_alb\_a()"/></td>

<td id="y\_text" class="hide\_div">Год:</td>

<td><input class="hide\_div" type="text" size="4" id="add\_y">

</tr>

<tr>

<td>Жанр:</td>

<td><input type="text" id="add\_gencomp"></td>

<td><input type="checkbox" name="4" id="ch\_gen"/></td>

</tr>

<tr><td><input type="button" id="add\_button" value="Добавить"></td></tr>

</table></div>

<div class="well"><table class="inv\_table1">

<tr><th colspan="3">Переименование данных</th></tr>

<tr>

<td><select id="upd\_obj">

<option value=""/>

<option value="ispolnitel">Исполнитель</option>

<option value="album">Альбом</option>

<option value="genre">Жанр</option>

<option value="composition">Композиция</option>

</select></td>

<td><input type="text" id="ren\_id" size='4' value='id'/></td>

<td><input type="text" id="ren\_name" value="Новое название" /></td>

<td><input type="button" id="ren\_button" value="Переименовать" onclick='ren\_func()'/></td>

</tr>

</table></div>

<div class="hide\_div" id="upl\_div">

<div class="well"><table class="search\_t">

<tr><th colspan="3">Добавить медиафайл к композиции</th></tr>

<tr>

<td><input type="text" id="upl\_id" size='4' value='id'/></td>

<td><input type="file" name="filename" id="file"></td>

<td><input type="button" onclick="upl\_func()" value="Загрузить"></td>

</tr>

</table></div>

</div>

<table class="search\_t">

<tr><td>Искать по</td>

<td><select id="adm\_search\_obj">

<option value="isp">Исполнителю</option>

<option value="alb">Альбомам</option>

<option value="gen">Жанрам</option>

<option value="comp">Композициям</option>

</select></td>

<td><input type="text" value="Поиск..." id="adm\_search\_text"/></td>

<td><input type="button" id="adm\_start\_search\_button" value="Вперед"/></td>

</tr>

</table>

</div>

<div class="hide\_div" id="search\_div">

<table class="search\_t">

<tr><td>Искать по</td>

<td><select id="search\_obj">

<option value="isp">Исполнителю</option>

<option value="alb">Альбомам</option>

<option value="gen">Жанрам</option>

<option value="comp">Композициям</option>

</select></td>

<td><input type="text" value="Поиск..." id="search\_text"/></td>

<td><input type="button" id="start\_search\_button" value="Вперед"/></td>

</tr>

</table>

</div>

<div id="dynamic\_content" class="hide\_div"> динамический контент<br></div>

</div>

</body>

</html>

<script>

$(document).ready(function(){

var chlogin=$.ajax({url:'includes/auth.php',method:'POST'});

chlogin.always(function(){

console.log(chlogin);

if (chlogin.responseText=="1") {

$("#index\_navigation").hide();

$("#admin\_navigation").show();

$("#user\_navigation").show();

}

if(chlogin.responseText=="0"){

$("#index\_navigation").hide();

$("#admin\_navigation").hide();

$("#user\_navigation").show();

}

});

});

</script>

<script src="script.js"></script>

<script>

audiojs.events.ready(function() {

var as = audiojs.createAll();

});

</script>

Стили CSS

html,body {

height:100% !important;

font-family: Arial, sans-serif;

font-size: 14px;

margin: 0 10% 0 !important;

}

body{border-left: 1px solid #FE9798;

border-right:1px solid #FE9798;}

td{padding-left: 0 !important;

padding-right: : 0 !important;}

.player{width:10%;

height:10%;}

#reg\_div{

background-color: #EEF;

border: 1px solid #FE9798;

padding-bottom: 7%;

padding-top: 7%;

padding-left: 30%;

}

#login\_div{

background-color: #EEF;

border: 1px solid #FE9798;

padding-bottom: 7%;

padding-top: 7%;

padding-left: 30%;

}

.well{margin-bottom: 1% !important;

padding: 2% !important;}

.inv\_table{width: 70%;}

.inv\_table1{width: 100%;}

.btn-link{

padding-top: 0 !important;

}

.search\_t{

width: 100%;

margin-top:1%;

}

.hide\_div{

display:none;

padding: 1%;

}

.btn-default{

margin-top: 10% !important;

margin-bottom: 0px !important;

}

#admin\_page{

padding: 3%;

}

#main{

min-height: 100%;

background: #F5DEB3;

}

#header {

background-image: url(http://beloweb.ru/wp-content/uploads/2012/08/ephemeral\_header03.jpg);

background-size: 100% 100%;

width: 100%;

height: 15%;

}

#conteiner{

top:20px;

bottom:20px;

min-height:100%;

}

#navigation {

background: #FE9798;

width: 100%;

height: 4%;

background: linear-gradient(#f9e8d0,#f0cd4e);

border-top: 1px solid #FE9798;

}

#user\_navigation, #admin\_navigation{ padding: 0; }

#content {

float: left;

min-height:81%;

width: 80%;

padding:0;

}

#menu{

border-top: 1px solid #FE9798;

float:left;

background: #F5DEB3;

width: 20%;

min-height:100%;

border-right: 1px solid #FE9798;

}

#main\_div{

border: 5%;

}

img{width: 80%;

border-radius: 30px;

display:block; margin:0 auto;}

#alb\_div{width:80px;}

ПРИЛОЖЕНИЕ Б

Основные фрагменты JavaScript-кода

Пример обработки нажатия кнопок бокового меню:

$("#all\_comp").click(function(){ *//КОМПОЗИЦИИ*

$("#dynamic\_content").hide(100);

show\_all("Композиция");

$("#main\_page\_div").hide("slow");

$("#personal\_page").hide("slow");

$("#admin\_page").hide("slow");

$("#search\_div").hide("slow");

});

Пример обработки нажатия кнопок навигации:

$("#main\_page").click(function(){ *//ГЛАВНАЯ СТРАНИЦА*

$("#main\_page\_div").show("slow");

$("#login\_div").hide("slow");

$("#reg\_div").hide("slow");

$("#admin\_page").hide("slow");

$("#personal\_page").hide("slow");

$("#search\_div").hide("slow");

});

Вывод данных из базы метаданных для пользователя:

function show\_all(nf){

$("#main\_page\_div").hide("slow");

$("#login\_div").hide("slow");

$("#reg\_div").hide("slow");

var show\_all=$.ajax({url:'includes/show\_all\_proc.php',method:'POST', data: {need\_find:nf}});

show\_all.always(function(){

console.log(show\_all);

if(show\_all.responseText=="1"){

$("#main\_page\_div").hide("slow");

$("#login\_div").show("slow");

$("#reg\_div").hide("slow");

}

else {

$("#dynamic\_content").show("slow");

$("#admin\_page").hide("slow");

$("#dynamic\_content").html(show\_all.responseText);}

});

};

Вывод данных из базы метаданных для администратора:

function show\_all\_for\_admin(){

var show\_all=$.ajax({url:'includes/show\_all\_for\_adm\_proc.php',method:'POST'});

show\_all.always(function(){

console.log(show\_all);

$("#dynamic\_content").hide(100);

$("#main\_page\_div").hide("slow");

$("#reg\_div").hide("slow");

$("#dynamic\_content").show("slow"); $("#dynamic\_content").html(show\_all.responseText);

});

};

Пример выполнения поиска по базе метаданных через раздел меню «Поиск»:

function search\_func(){

var obj;

switch(document.getElementById("search\_obj").selectedIndex){

case 0:

obj="1";

break;

case 1:

obj="2";

break;

case 2:

obj="3";

break;

case 3:

obj="4";

break;

}

var search=$.ajax({url:'includes/search\_proc.php',method:"POST", data:{keyword:$("#search\_text").val(),obj:obj}});

search.always(function(){

console.log(search);

$("#dynamic\_content").show("slow"); $("#dynamic\_content").html(search.responseText);

});

};

Функция переименования данных в базе метаданных:

function ren\_func(){

var i=+document.getElementById("upd\_obj").selectedIndex

var code=$("#ren\_id").val();

var name=$("#ren\_name").val();

var ren=$.ajax({url:'includes/ren\_func.php',method:'POST',data:{code:code, name:name,i:i}});

ren.always(function(){

console.log(ren);

show\_all\_for\_admin();

alert(ren.responseText);

});

};

Функция добавления в избранное:

function like(code){

var i=0;

var like=$.ajax({url:'includes/like\_dislike.php', method:'POST', data:{idc:code, a:i}})

like.always(function(){

console.log(like);

show\_all("Композиция");

});

};

Функция удаления из избранного:

function dislike(code){

var i=1;

var like=$.ajax({url:'includes/like\_dislike.php', method:'POST', data:{idc:code, a:i}})

like.always(function(){

console.log(like);

show\_all("Композиция");

});

};

Функция добавления медиаданных:

function upl\_func(){

var um=$.ajax({url:'includes/upd\_media.php', method:'POST', data:{code:$("#upl\_id").val()}});

um.always(function(){

console.log(um);

});

var data = new FormData();

$.each( files, function( key, value ){data.append( key, value );

});

$.ajax({

url: './submit.php?uploadfiles',

type: 'POST',

data: data,

cache: false,

dataType: 'json',

processData: false,

contentType: false,

success: function( respond, textStatus, jqXHR ){

if( typeof respond.error === 'undefined' ){

var files\_path = respond.files;

var html = '';

$.each( files\_path, function( key, val ){ html += val +'<br>'; } )

show\_all\_for\_admin();

}

else{

console.log('ОШИБКИ ОТВЕТА сервера: ' + respond.error );

}

},

error: function( jqXHR, textStatus, errorThrown ){

console.log('ОШИБКИ AJAX запроса: ' + textStatus );

}

});

};

ПРИЛОЖЕНИЕ В

Основные фрагменты PHP-кода

Скрипт процесса авторизации:

<?

require\_once("connection.php");

if(($\_POST["login\_name"]=="")||($\_POST["login\_pass"]=="")) {echo("Введите все необходимые данные!");}

else {

$user=$db->query("CALL GetPass('{$\_POST['login\_name']}')");

$user\_data = $user->fetch\_array();

while($db->more\_results()){

$db->next\_result();

}

if(($user\_data["ЛогинПользователя"]==$\_POST["login\_name"])&&($user\_data["Пароль"]==$\_POST["login\_pass"]))

{

session\_start();

$\_SESSION["name"]=$\_POST["login\_name"];

$\_SESSION["password"]=$\_POST["login\_pass"];

$\_SESSION["role"]=$user\_data["КодРоли"];

echo($user\_data["КодРоли"]);

}

else {echo("Error");}

}

?>

Скрипт процесса регистрации:

<?

require\_once("connection.php");

if(($\_POST["reg\_name"]=="")||($\_POST["reg\_pass"]=="")) {echo("Введите все необходимые данные!");}

else {

$user=$db->query("CALL GetPass('{$\_POST['reg\_name']}')");

$user\_data = $user->fetch\_array();

while($db->more\_results()){

$db->next\_result();

}

if($user\_data["ЛогинПользователя"]==$\_POST["reg\_name"]) {echo("Данный логин уже занят!");}

else

{

$u=$db->query("INSERT INTO `Пользователь`(`ЛогинПользователя`, `Пароль`) VALUES ('{$\_POST["reg\_name"]}','{$\_POST["reg\_pass"]}')");

}

}

?>

Скрипт процесса добавления данных в базу метаданных:

<?

require\_once("connection.php");

if($\_POST["i"]=="0"){

$db->query("INSERT INTO Исполнитель (НазваниеИсполнителя) VALUES('{$\_POST['ispn']}')");

$user=$db->query("SELECT MAX(КодИсполнителя)AS КодИсполнителя FROM Исполнитель");

$user\_data = $user->fetch\_array();

while($db->more\_results()){

$db->next\_result();

}

$ic=+$user\_data["КодИсполнителя"];

} else {$ic=+$\_POST["ispn"];}

if($\_POST["a"]=="0"){

$db->query("INSERT INTO Альбом (НазваниеАльбома,Год) VALUES('{$\_POST['an']}','{$\_POST['y']}')");

$user=$db->query("SELECT MAX(КодАльбома)AS КодАльбома FROM Альбом");

$user\_data = $user->fetch\_array();

while($db->more\_results()){

$db->next\_result();

}

$ac=+$user\_data["КодАльбома"];

} else {$ac=+$\_POST["an"];}

if($\_POST["g"]=="0"){

$db->query("INSERT INTO Жанр (НазваниеЖанра) VALUES('{$\_POST['gn']}')");

$user=$db->query("SELECT MAX(КодЖанра)AS КодЖанра FROM Жанр");

$user\_data = $user->fetch\_array();

while($db->more\_results()){

$db->next\_result();

}

$gc=+$user\_data["КодЖанра"];

} else {$gc=+$\_POST["gn"];}

$db->query("INSERT INTO Композиция (КодИсполнителя,КодАльбома,КодЖанра,НазваниеКомпозиции) VALUES ('{$ic}','{$ac}','{$gc}','{$\_POST['cn']}')");

?>

Скрипт удаления медиаданных:

<?

require\_once("connection.php");

$code=$\_POST["code"];

$user=$db->query("SELECT Медиа FROM Композиция WHERE КодКомпозиции='{$code}'");//запрос к бд

$user\_data = $user->fetch\_array();

while($db->more\_results()){

$db->next\_result();

}

$filename=$user\_data["Медиа"];

unlink($filename);

$db->query("UPDATE Композиция SET Медиа='' WHERE КодКомпозиции='{$code}'");

?>

Скрипт переименования данных из базы метаданных:

<?

require\_once("connection.php");

switch($\_POST['i']){

case 4:

$db->query("UPDATE Композиция SET НазваниеКомпозиции='{$\_POST['name']}' WHERE Композиция.КодКомпозиции='{$\_POST['code']}'");

echo("comp");

break;

case 1:

$db->query("UPDATE Исполнитель SET НазваниеИсполнителя='{$\_POST['name']}' WHERE Исполнитель.КодИсполнителя='{$\_POST['code']}'");

break;

case 2:

$db->query("UPDATE Альбом SET НазваниеАльбома='{$\_POST['name']}' WHERE Альбом.КодАльбома='{$\_POST['code']}'");

break;

case 3:

$db->query("UPDATE Жанр SET НазваниеЖанра='{$\_POST['name']}' WHERE Жанр.КодЖанра='{$\_POST['code']}'");

break;

}

?>

Скрипт редактирования списка избранного:

<?

require\_once("connection.php");

session\_start();

$idc=+$\_POST['idc'];

if($\_POST['a']==1){

$r=$db->query("DELETE FROM Плейлист WHERE ЛогинПользователя='{$\_SESSION['name']}' AND КодКомпозиции='{$idc}'");

}

else {

$r2=$db->query("INSERT INTO Плейлист (ЛогинПользователя,КодКомпозиции) VALUES('{$\_SESSION['name']}','{$idc}')");

}

?>

Скрипт выполнения поиска по базе метаданных:

<?

require\_once("connection.php");

session\_start();

$keyword1=$\_POST['keyword'];

$keyword1 = trim($keyword1); *// убираются лишние пробелы из начала и конца строки*

$keyword1 = stripslashes($keyword1); *//удаляет экранирование символов*

$keyword1 = htmlspecialchars($keyword1); *// заменяет html теги на коды*

$keyword1=strtoupper($keyword1);

$s="%";

$keyword=$s.$keyword1.$s;

switch($\_POST['obj']){

case "1":

$result=$db->query("SELECT \* from Композиция JOIN Альбом ON Композиция.КодАльбома=Альбом.КодАльбома JOIN Исполнитель ON Композиция.КодИсполнителя=Исполнитель.КодИсполнителя JOIN Жанр ON Композиция.КодЖанра=Жанр.КодЖанра WHERE Исполнитель.НазваниеИсполнителя LIKE '{$keyword}'");

break;

case "2":

$result=$db->query("SELECT \* from Композиция JOIN Альбом ON Композиция.КодАльбома=Альбом.КодАльбома JOIN Исполнитель ON Композиция.КодИсполнителя=Исполнитель.КодИсполнителя JOIN Жанр ON Композиция.КодЖанра=Жанр.КодЖанра WHERE Альбом.НазваниеАльбома LIKE '{$keyword}'");

break;

case "3":

$result=$db->query("SELECT \* from Композиция JOIN Альбом ON Композиция.КодАльбома=Альбом.КодАльбома JOIN Исполнитель ON Композиция.КодИсполнителя=Исполнитель.КодИсполнителя JOIN Жанр ON Композиция.КодЖанра=Жанр.КодЖанра WHERE Жанр.НазваниеЖанра LIKE '{$keyword}'");

break;

case "4":

$result=$db->query("SELECT \* from Композиция JOIN Альбом ON Композиция.КодАльбома=Альбом.КодАльбома JOIN Исполнитель ON Композиция.КодИсполнителя=Исполнитель.КодИсполнителя JOIN Жанр ON Композиция.КодЖанра=Жанр.КодЖанра WHERE Композиция.НазваниеКомпозиции LIKE '{$keyword}'");

break;

}

$nr=$result->num\_rows;

if($nr==0){echo("Ничего не найдено");}

else{

$user\_data = $result->fetch\_all(MYSQLI\_ASSOC);

while($db->more\_results()){

$db->next\_result();

}

$name=$\_SESSION['name'];

$favs=$db->query("SELECT КодКомпозиции FROM Плейлист WHERE Плейлист.ЛогинПользователя='{$name}'");

$user\_favs = $favs->fetch\_all(MYSQLI\_ASSOC);

if($user\_favs==NULL){$user\_favs1=[0];}

$ids = [];

foreach ($user\_favs as $value) {

array\_push($ids, $value["КодКомпозиции"]);

}

while($db->more\_results()){

$db->next\_result();

}

echo('<br><table class="table table-striped">');

echo("<tr><th>Название</th><th>Исполнитель</th><th>Альбом</th><th colspan='2'>Жанр</th></tr>");

foreach ($result as $r)

{

$comp=substr($r["НазваниеКомпозиции"],0,25); $isp=substr($r["НазваниеИсполнителя"],0,25);

$alb=substr($r["НазваниеАльбома"],0,25);

$gen=substr($r["НазваниеЖанра"],0,25);

$code=substr($r["КодКомпозиции"],0);

if (in\_array($code, $ids)) { echo("<tr><td>".$comp."</td><td>".$isp."</td><td>".$alb."</td><td>".$gen."</td><td><input type='button' class='dislike\_button' value='&#128148' onclick='dislike\_s(".$code.")'></td></tr>");

}

else { echo("<tr><td>".$comp."</td><td>".$isp."</td><td>".$alb."</td><td>".$gen."</td><td><input type='button' class='like\_button' value='&#10084' onclick='like\_s(".$code.")'></td></tr>");

}

}

echo("</table><br>");

}

?>

Скрипт вывода данных из базы метаданных:

<?

require\_once("connection.php");

session\_start();

if(isset($\_SESSION["name"]) && $\_SESSION["name"]!="")

{

$need\_find=$\_POST["need\_find"];

$result=$db->query("Call Show\_all");

$user\_data = $result->fetch\_all(MYSQLI\_ASSOC);

while($db->more\_results()){

$db->next\_result();

}

$name=$\_SESSION['name'];

$favs=$db->query("SELECT КодКомпозиции FROM Плейлист WHERE Плейлист.ЛогинПользователя='{$name}'");

$user\_favs = $favs->fetch\_all(MYSQLI\_ASSOC);

if($user\_favs==NULL){$user\_favs1=[0];}

$ids = [];

foreach ($user\_favs as $value) {

array\_push($ids, $value["КодКомпозиции"]);

}

while($db->more\_results()){

$db->next\_result();

}

echo('<br><table class="table table-striped">');

switch($\_POST["need\_find"])

{

case "Композиция":

echo("<tr><th>Название</th><th>Исполнитель</th><th>Альбом</th><th>Жанр</th><th></th></tr>");

break;

case "Исполнитель":

echo("<tr><th>Исполнитель</th><th>Композиция</th><th>Жанр</th></tr>");

break;

case "Альбом":

echo("<tr><th>Название</th><th>Исполнитель</th><th>Жанр</th><th>Год</th></tr>");

break;

case "Жанр":

echo("<tr><th>Название</th><th>Исполнитель</th></tr>");

break;

}

foreach ($result as $r)

{

switch($\_POST["need\_find"])

{

case "Композиция":

$comp=substr($r["НазваниеКомпозиции"],0,25);

$isp=substr($r["НазваниеИсполнителя"],0,25);

$alb=substr($r["НазваниеАльбома"],0,25);

$gen=substr($r["НазваниеЖанра"],0,25);

$code=substr($r["КодКомпозиции"],0);

if (in\_array($code, $ids)) {

echo("<tr><td>".$comp."</td><td>".$isp."</td><td>".$alb."</td><td>".$gen."</td><td><input type='button' class='dislike\_button' value='&#128148' onclick='dislike(".$code.")'></td></tr>");

}

else {

echo("<tr><td>".$comp."</td><td>".$isp."</td><td>".$alb."</td><td>".$gen."</td><td><input type='button' class='like\_button' value='&#10084' onclick='like(".$code.")'></td></tr>");

}

break;

case "Исполнитель":

$comp=substr($r["НазваниеКомпозиции"],0,25);

$isp=substr($r["НазваниеИсполнителя"],0,25);

$gen=substr($r["НазваниеЖанра"],0,25);

echo("<tr><td>".$isp."</td><td>".$comp."</td><td>".$gen."</td></tr>");

break;

case "Альбом":

$isp=substr($r["НазваниеИсполнителя"],0,25);

$alb=substr($r["НазваниеАльбома"],0,25);

$gen=substr($r["НазваниеЖанра"],0,25);

$y=substr($r["Год"],0,25);

echo("<tr><td>".$alb."</td><td>".$isp."</td><td>".$gen."</td><td>".$y."</td></tr>");

break;

case "Жанр":

$isp=substr($r["НазваниеИсполнителя"],0,25);

$gen=substr($r["НазваниеЖанра"],0,25);

echo("<tr><td>".$gen."</td><td>".$isp."</td></tr>");

break;

}

}

echo("</table><br>");

} else

{echo("1");}

?>

Скрипт отображения плейлиста:

<?

require\_once("connection.php");

session\_start();

$user\_login=$\_SESSION["name"];

echo("<h2>Добро пожаловать, {$user\_login}!</h2><br><audio src='' preload='auto' /><br>");

$result=$db->query("Call Show\_playlist('{$user\_login}')");

$user\_data = $result->fetch\_all(MYSQLI\_ASSOC);

echo('<br><table class="table table-striped"><tr><th>Название</th><th>Исполнитель</th><th>Альбом</th><th colspan="3">Жанр</th></tr>');

foreach ($result as $r)

{

$comp=substr($r["НазваниеКомпозиции"],0,25);

$isp=substr($r["НазваниеИсполнителя"],0,25);

$alb=substr($r["НазваниеАльбома"],0,25);

$gen=substr($r["НазваниеЖанра"],0,25);

$media=substr($r["Медиа"], 0);

if ($media!=NULL){

echo("<tr><td>".$comp."</td><td>".$isp."</td><td>".$alb."</td><td>".$gen."</td><td><input type='button' class='dislike\_button' value='&#128148' onclick='dislike\_pp(".$code.")'></td><td><input type='button' class='play\_button' onclick='play\_button\_func(".$media.")' value='&#127926'/></td></tr>");}

else {echo("<tr><td>".$comp."</td><td>".$isp."</td><td>".$alb."</td><td>".$gen."</td><td><input type='button' class='dislike\_button' value='&#128148' onclick='dislike\_pp(".$code.")'></td><td></td></tr>");}

}

echo("</table><br><script>

audiojs.events.ready(function() {

var as = audiojs.createAll();

});

</script>");

?>