ПРИДНЕСТРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. Т.Г. ШЕВЧЕНКО
Рыбницкий филиал
Кафедра информатики и программной инженерии

КУРСОВАЯ РАБОТА

по дисциплине
«ПРОГРАММИРОВАНИЕ»
на тему:
«РАЗРАБОТКА ИНФОРМАЦИОННО-ПОИСКОВОЙ СИСТЕМЫ»

Студентки I курса
направления «Программная инженерия»
профиля «Разработка программно–
информационных систем»
Нихолат Александры Эдуардовны

Научный руководитель:
ст. преподаватель
Гарбузняк Елена Сергеевна

Рыбница, 2022

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В связи с эпидемиологической ситуацией и политической ситуацией в мире возросло значение и необходимость дистанционного обучения. Однако есть как студенты, так и преподаватели с малым опытом в использовании компьютерных технологий, а технологические навыки являются обязательными для успешного прохождения дистанционного обучения. Поэтому возникла необходимость организовать обучение использованию электронных курсов.

Актуальность исследования: чтобы помочь пользователю в упрощенном и быстром поиске видео-инструкций по сайту «Видео*Pi*», их сортировке и просмотру была создана информационно-поисковая система.

Объектом исследования является информационно-поисковые системы.

Предметом исследования являются возможности *php* при подключении к базе данных (БД) для создания информационно-поисковой системы (ИПС).

Целью исследования: разработать информационно-поисковую систему для образовательного сайта.

Исходя из цели исследования, были поставлены следующие **задачи:**

- исследовать понятие и виды информационно-поисковой системы;
- проанализировать структуру информационно-поисковой системы;
- изучить элементы информационно-поисковой системы;
- рассмотреть виды существующих программных продуктов;
- выбрать среду программирования;
- создать структуру базы данных и организовать работу с БД;
- провести тестирование программы;
- выполнить анализ полученных результатов.

Цели, задачи, объект и предмет исследования определили **структуру** работы. Она состоит из введения, двух глав, заключения и списка литературы.

В первом разделе раскрыты понятие и виды ИПС, сущность БД и система управления базами данных (СУБД), понятие релевантности, индексирования и поискового запроса, структура ИПС, обзор существующих программных продуктов и выбранной среды программирования.

Второй раздел посвящен разработке программного продукта. В нём описываются этапы разработки, структура пользовательских подпрограмм, тестирование программы и анализ полученных результатов.

В заключении подведены итоги и обобщаются результаты исследования, преимущества и недостатки реализованного программного продукта, а также перспективы развития.

В приложениях представлены блок-схема, описывающая структуру программы, и листинг разработанного программного продукта.

1. ИССЛЕДОВАНИЕ И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Понятие и виды информационно-поисковой системы

Информационно-поисковая система – это система, обеспечивающая поиск и отбор необходимых данных в специальной базе с описаниями источников информации (индексе) на основе информационно-поискового языка и соответствующих правил поиска [1-5].

В число составных частей конкретной информационно-поисковой системы, кроме информационно-поискового языка, правил перевода и критерия соответствия, входят также средства ее технической реализации, массив текстов (документов), в котором осуществляется информационный поиск.

Программной частью поисковой системы является поисковая машина (поисковый движок) – комплекс программ, обеспечивающий функциональность поисковой системы.

Поисковый запрос – исходная информация для осуществления поиска с помощью поисковой системы.

Запросы, которые формируют пользователи информационных систем, реализуются следующими способами:

- сообщения, являющиеся ответом на запрос, хранятся в явном виде в базе данных, и процесс получения ответа представляет собой выделение подмножества знаний из файлов базы данных, удовлетворяющих запросу;
- ответ не существует в явном виде в базе данных и формируется в процессе логического вывода на основании имеющихся данных.

Каждая ИПС состоит из двух частей: базы данных и системы управления базами данных.

База данных – это поименованная совокупность структурированных данных, относящихся к определенной предметной области.

Система управления базами данных – это комплекс программных и языковых средств, необходимых для создания баз данных, поддержания их в актуальном состоянии и организации сортировки, поиска в них необходимой информации.

Главной задачей любой ИПС является поиск информации релевантным информационным потребностям пользователя. Очень важно в результате проведенного поиска найти все документы, относящиеся к запросу и ничего лишнего. Поэтому вводится качественная характеристика процедуры поиска – релевантность.

Релевантность – степень соответствия найденного документа или набора документов информационным нуждам пользователя.

ИПС бывают двух типов:

1. Документографические. В документографических ИПС каждому документу присваивается индивидуальный код, составляющий поисковый образ. Поиск идет не по самим документам, а по их поисковым образам.

2. Фактографические. В фактографических ИПС хранятся не документы, а факты, относящиеся к какой–либо предметной области. Поиск осуществляется по образцу факта.

1.2. Структура информационно-поисковой системы

В основу построения структуры информационно-поисковой системы легло её функциональное назначение, область применения и особенности описываемой ею предметной области [6].

Функционально ИПС предназначена для быстрого и удобного поиска и выборки данных из больших массивов информации по шаговым двигателям как для внутренней работы с данными, так и для подготовки их для различных систем автоматизированного проектирования (САПР). Это накладывает определённые требования на построение пользовательского интерфейса и на форму предоставления информации.

Реализация вышеперечисленных требований возложена на следующий ряд структурных компонентов, так называемых блоков: проверки БД на целостность; просмотра; редактирования; защиты паролем; поиска; вывода результата; хранения параметров поиска; помощи.

В основе выбора именно такой структуры информационно-поисковой системы по шаговым двигателям лежит очень простая логика – любой блок системы должен получать данные, обрабатывать их и выдавать пользователю в определенном порядке, обеспечивая логику процесса.

Рассмотрим каждый блок более подробно (рис. 1.1):

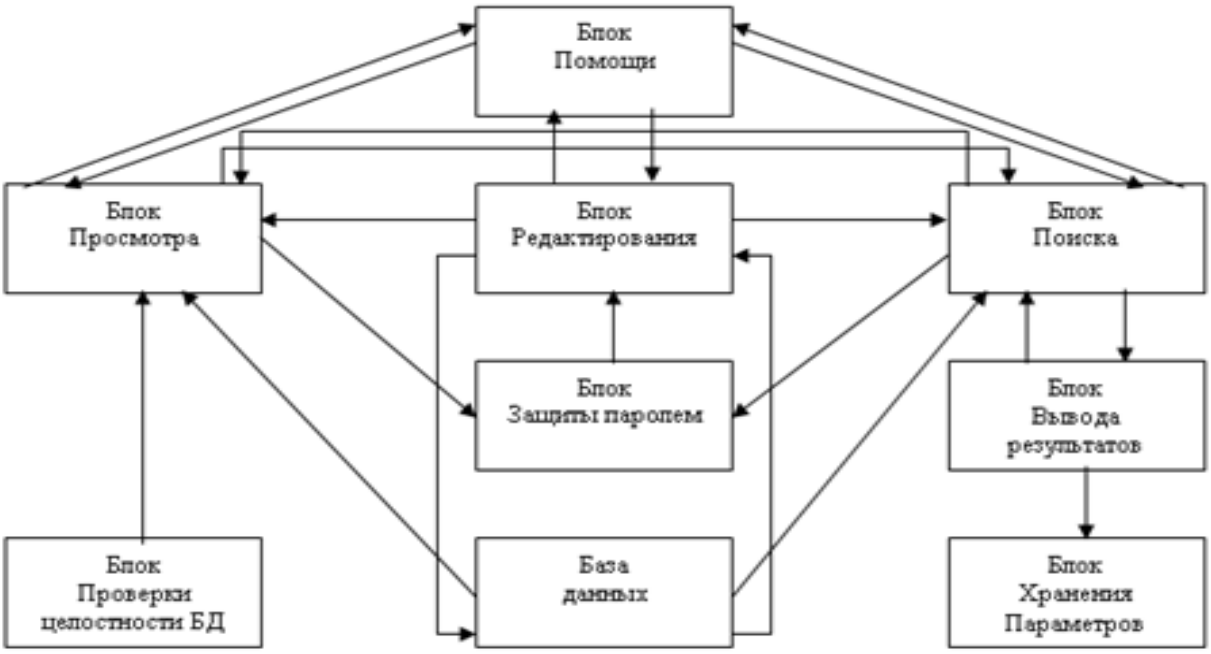


Рис. 1.1. Структура ИПС

Блок просмотра позволяет начать работу в системе с просмотра БД и далее выбрать другой режим работы.

Блок редактирования производит редактирование только числовых полей БД и позволяет изменять характеристики, вводить новые и удалять старые записи в таблицы БД. Здесь также можно произвести смену режима работы.

Блок защиты паролем осуществляет блокировку доступа к редактированию данных путем ввода шестизначного пароля. Блок поиска предназначен для осуществления поиска по введенному техническому заданию (ТЗ) и перехода к другим режимам работы. Блок вывода результатов поиска выводит на экран в определенном порядке все найденные шаговые двигатели и их характеристики в соответствии с ТЗ поиска. Блок хранения параметров поиска записывает и хранит информацию до следующего этапа поиска.

Блок помощи выполняет роль подсказки в различных режимах работы системы. Область применения ИПС, как было указано выше, – это внутренняя работа с информацией и обработка информации для использования её в работе САПР, включающей в свой состав ИПС как один из модулей. Из этого вытекают очень высокие требования к надёжности функционирования системы, поскольку любая САПР – это достаточно сложное построение с заданными параметрами надежности, по крайней мере, не меньшей, чем вся система в целом.

1.3. Элементы информационно-поисковой системы

Информационно-поисковый язык – знаковая система, предназначенная для описания (путём индексирования) основного смыслового содержания текстов (документов) или их частей, а также для выражения смыслового содержания информационных запросов с целью реализации информационного поиска. Любой абстрактный информационно - поисковый язык (ИПЯ) состоит из алфавита (списка элементарных символов), правил образования и правил интерпретации. Правила образования устанавливают, какие комбинации элементарных символов допускаются при построении слов и выражений, а правила интерпретации – как надлежит понимать эти слова и выражения [7].

Индексирование – процесс выражения главного предмета или темы текста какого-либо документа в терминах информационно-поискового языка. Применяется для облегчения поиска необходимого текста среди множества других.

ИПЯ должен располагать лексико-грамматическими средствами, необходимыми для выражения основного смыслового содержания любого текста и смысла любого информационного запроса по данной отрасли или предмету, быть недвусмысленным (допускать одно истолкование каждой записи), удобным для алгоритмического сопоставления и отождествления (полного или частичного) записей основного смыслового содержания текстов и смыслового содержания информационных запросов. При разработке конкретного ИПЯ учитываются специфика отрасли или предмета, для которой этот язык создаётся, особенности текстов, образующих поисковый массив, характер информационных потребностей, для удовлетворения которых создается данная информационно - поисковая система.

Отобранные из естественного языка слова и словосочетания, в совокупности образующие основной словарный состав, служат как бы алфавитом данного ИПЯ. Правила образования в таких ИПЯ выполняют функцию синтаксиса.

Дескриптор – лексическая единица (слово, словосочетание) информационно-поискового языка, служащая для описания основного смыслового содержания документов. Дескрипторы служат также для формулировки информационных запросов при поиске документов в информационно-поисковой системе

Центральной частью каждой ИПС является информационно-поисковый массив (ИПМ), который может быть организован различными способами. В документальном информационно-поисковом языке (ДИПС) ИПМ подразделяется на две части: сами документы или их копии и поисковый образ документа (ПОД) с адресами – номерами документов в поисковом массиве. Поиск осуществляется по второй части ИПМ [8].

В полнотекстовых базах данных поиск может осуществляться как по самому тексту документа (при такой организации поиск будет очень медленным), так и в специальных поисковых файлах, содержащих информацию о тексте документа (индексах).

В ИПС используют две принципиальные схемы информационного массива – прямую и инверсную.

При прямой организации каждому документу соответствует перечень слов, составляющих текст или поисковый образ документа. При этом отыскание нужных документов производится путем поочередного сравнения поисковых образов документов со словами, составляющими информационный запрос, т.е. реализуется принцип последовательного доступа к данным. Достоинства прямой схемы состоят в простоте организации и использования, например для включения нового документа в массив достаточно добавить новую запись в файл.

Недостатком прямой организации поиска является необходимость последовательного просмотра ПОД всех документов, что ведет к большим затратам времени, поэтому в современных текстовых базах данных применяют инверсный способ.

Инверсный способ организации поискового массива предусматривает создание инвертированной матрицы, в которой и происходит поиск (ее называют инвертированным матричным индексом).

Правила индексирования, следуя которым производится описание средствами ИПЯ документов и запросов (перевод их с естественного языка на информационно-поисковый). В результате индексирования документа получается ПОД, а индексирования запроса – поисковое предписание [9].

Если указан только атрибут, то индексы не создаются. Сервер каталогов поддерживает следующие правила индексации:

- равенство;
- порядок;
- приблизительное равенство;

- подстрока;
- обратный.

Указание правила индексации для атрибута позволяют управлять созданием и обслуживанием специальных индексов значений атрибутов. Таким образом, удается существенно сократить время отклика при выполнении поиска с фильтрами, включающими эти атрибуты.

Критерий соответствия – совокупность правил, по которым при информационном поиске определяется степень соответствия поискового образа документа поисковому предписанию и принимается решение о выдаче или невыдаче этого документа в ответ на информационный запрос. Наряду с информационно-поисковым языком критерии соответствия являются одним из элементов информационно-поисковой системы.

Обслуживающий персонал – индексаторы и технические работники обеспечивают обработку и ввод в систему документов, а также операторы ИПС производят поиск информации и выдачу ее потребителю (в качестве оператора ИПС может выступать и сам потребитель информации).

1.4. Обзор методов решений подобных задач

Одной из самых популярных и востребованных поисковых систем в России на сегодняшний день является поисковик «Яндекс» (рис. 1.2) [12].

Она поддерживает собственный каталог Интернет-ресурсов. Также является лучшей поисковой системой для выявления иллюстраций. Обладает развернутой системой формирования запроса. В частности, допускается ввод поискового предписания на естественном языке – в этом случае все необходимые расширения производятся автоматически [11].

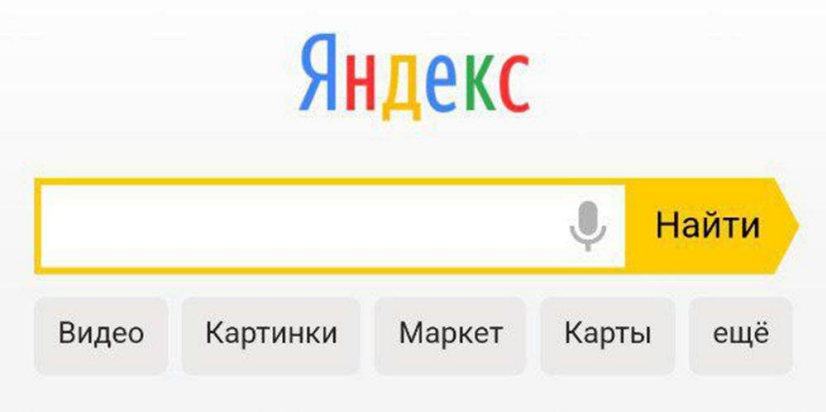


Рис. 1.2. Поисковая система «Яндекс»

Преимуществом «Яндекса» перед другими поисковиками в российском интернете стала возможность осуществления поиска информации по регионам. Программа выводит на экран результаты, которые являются наиболее релевантными для пользователя в соответствии с его местоположением. Аналогичный алгоритм был со временем введен и в *Google.*

Другим преимуществом «Яндекс» является быстрая работа службы поддержки.

Yandex браузер создан на базе *Chromium.* Характерным для него является турбоускоритель загрузки страниц, умная адресная строка, возможность перевода иноязычных сайтов.

Неудобством поисковика является частое несоответствие результатов поиска созданному запросу. Для достижения более точного результата человеку приходится использовать дополнительные поисковые операторы и применять собственные приемы для преобразования словосочетаний таким образом, чтобы система вывела более подходящие данные.

Google является самым посещаемым сайтом в мире (рис. 1.3) [13].

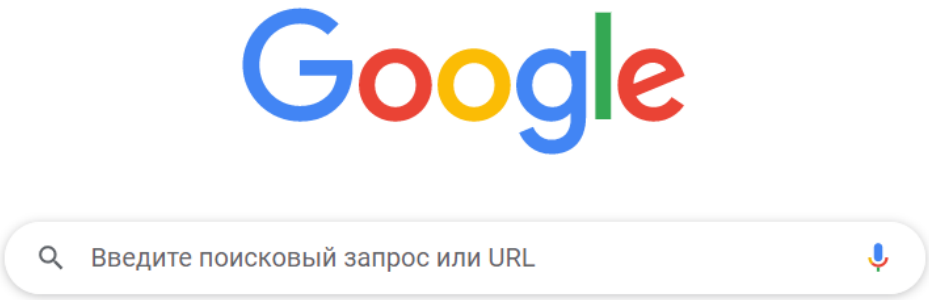


Рис. 1.3. Поисковая система «*Google*»

Основным преимуществом поисковой системы *Google* все называют простоту ее использования. Среди прочих положительных качеств можно отметить:

- отсутствие необходимости использования особых знаков при введении ключевых слов для поиска;
- наличие огромной базы данных;
- высокая скорость работы;
- максимальная точность информации – полное соответствие результатов поиска запросу;
- учет качества страниц при поиске, а не только их количества.

При всех своих огромных преимуществах *Google* не лишен и ряда недостатков, среди которых:

- постоянное появление в результатах поиска ссылок на сайты с устаревшей и никому не нужной информацией, вытекающее из принципа вечного хранения информации;
- появление в поиске сайтов, находящихся в разработке, что также вытекает из достоинства *Google* – его скорости индексации сайтов;
- сложность поиска по абстрактным запросам;
- слишком большой объем информации при широких запросах, так как результаты поиска не объединяются в категории.

Google использует показатель *PageRank* найденных по запросу страниц, чтобы определить порядок выдачи этих страниц посетителю в

результатах поиска. Например, если один Интернет-ресурс имеет несколько страниц полезной пользователю информации, поисковая система и выведет несколько результатов с данного сайта.

В мае 2005 года была создана компания *mail.ru Group*. Она продвигает общую интегрированную площадку коммуникационных и развлекательных интернет-сервисов.

В ноябре 2013 в *Google Play* появилась новая версия поискового приложения от компании *Mail.Ru* (рис. 1.4), позволяющего переходить с главного экрана в любые социальные сети и содержащего быстрый доступ к поиску по картинам, видео и новостям. Ее характерная особенность заключаются в возможности поиска по социальным сетям [14].

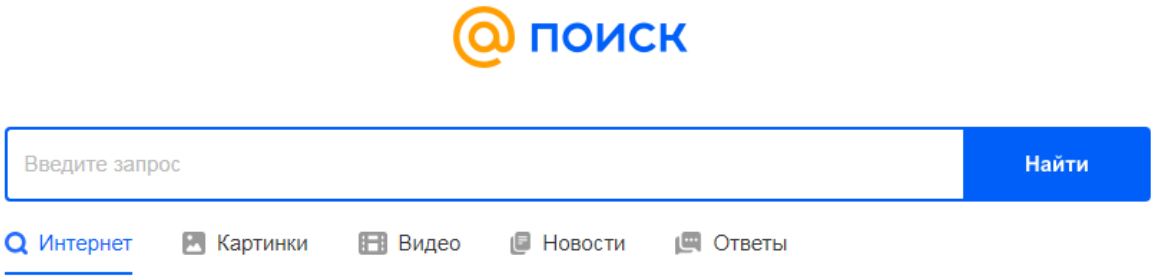


Рис. 1. 4. Система поиска «*Mail.ru*»

DuckDuckGo – это довольно известная поисковая система с открытым исходным кодом (рис. 1.5) [15].

Рис. 1.5. Поисковая система «*DuckDuckGo*»

Кроме собственного робота, поисковик использует результаты других источников: *Yahoo, Bing*, «Википедии».

DuckDuckGo позиционирует себя как поиск, обеспечивающий максимальную приватность и конфиденциальность. Система не собирает никаких данных о пользователе, не хранит логи (нет истории поиска), использование файлов *cookie* максимально ограничено.

DuckDuckGo формирует объективную картину, не зависящую от вашего прошлого поведения в Сети, и избавляет от тематической рекламы *Google* и «Яндекса», основанной на ваших запросах [10]. Также здесь минимум рекламы и чистый удобный интерфейс. Среди недостатков стоит отметить отсутствие голосового поиска и поиска по изображениям. Еще одна интересная функция, что эта поисковая система имеет – это нулевой клик информации, где все ответы представлены на первой странице, однако данные весьма ограничены.

1.5. Выбор среды программирования

Для реализации поставленной задачи, а именно для создания информационно-поисковой системы существует достаточно много сред разработки. Среди таких сред возможность реализовать создание программы есть у *XAMPP и Visual Studio Code.*

Microsoft Visual Studio – линейка продуктов компании *Microsoft*, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов [16].

Важным фактором в выборе среды разработке послужил удобный многоязычный интерфейс пользователя. *Visual Studio* включает в себя редактор исходного кода с поддержкой технологии *IntelliSense* и возможностью простейшего рефакторинга кода. Он поддерживает ряд языков программирования, подсветку синтаксиса, отладку, навигацию по коду, другие возможности. Встроенный отладчик может работать как отладчик

уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных.

XAMPP – кроссплатформенная сборка локального веб-сервера, содержащая *Apache, MariaDB*, интерпретатор скриптов *PHP,* язык программирования *Perl* и большое количество дополнительных библиотек, позволяющих запустить полноценный веб-сервер.

Полный пакет содержит:

- *Web*-сервер *Apache* с поддержкой *SSL*;
- СУБД *MySQL*;
- *PHP;*
- *Perl;*
- *FTP*-сервер *FileZilla;*
- *POP3*/*SMTP* сервер;
- утилиту *phpMyAdmin*.

Программа свободно распространяется согласно лицензии и является бесплатным, удобным в работе *web*-сервером, способным обслуживать динамические страницы. Пользовательский интерфейс программы настолько прост, что ее называют «сборкой для ленивых». Также программа поддерживает создание и управление базами данных *MySQL* и *SQLite*.

После изучения теоретического материала было решено создать программный продукт, реализующий информационно-поисковую систему, аналогичную поисковым системам во Всемирной паутине, в среде программирования *Visual Studio Code* с помощью сборки *Web*-серверов *XAMPP*.

2. РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА. ОПИСАНИЕ РЕЗУЛЬТАТОВ

2.1. Постановка задачи и системные требования

Для реализации программного продукта были поставлены задачи:

1. Выбор БД.
2. Формирование БД (разработка структуры, подключение).
3. Организация записи данных в БД;
4. Обращение к БД.

Системные требования. *XAMPP* – это один из немногих комплектов для разработчика, который будет хорошо работать как на *Windows*, так и на *Linux, Mac OS X, Solaris*. Для ОС *Windows*, для которой существует две версии *XAMPP Windows* и *XAMPP Lite*.

Системные требования к *XAMPP Windows:*

- 128 *MB RAM*;
- 320 *MB* свободного места на жестком диске;
- *Windows* 2000, *XP* (*Server* 2003), *Vista* (*Server* 2008), 7;
- все 32 *bit* ОС (64 должны работать).

Lite отличается отсутствием некоторых библиотек, и соответственно занимает меньше места на диске, и сам установочный файл меньше весит.

Системные требования к *XAMPP Lite:*

- 28 *MB RAM;*
- 192 *MB* свободного места на жестком диске;
- *Windows* 2000, *XP* (*Server* 2003), *Vista* (*Server* 2008), 7;
- все 32 *bit* ОС (64 должны работать).

Для работы с БД сначала нужно создать пользователя БД и пароль. Чтобы упростить работу с базами данных *MySQL* можно установить специальный набор скриптов *phpMyAdmin. phpMyAdmin* представляет интуитивный веб-интерфейс для управления базами данных *MySQL*.

2.2. Работа с базой данных

Базы данных (*Database, db*) широко применяются для структурирования и хранения информации. Они важны для успешной работы современных *web*-сайтов, причем используются для хранения данных и на тех сайтах, которые написаны на *PHP*. Таким образом, на практике *databases* применяют [17]:

- для сохранения пользовательских данных, полученных с помощью форм регистрации;
- для организации поиска по страницам сайта;
- для хранения комментариев;
- для хранения всевозможного контента (статей, изображений, медиафайлов) и многих других данных (*data*).

Для создания информационно-поисковой системы с использованием базы данных необходимо разработать структуру БД. Создание базы данных требует работы в программном обеспечении для локальной разработки и отладки сайтов *XAMPP*. При создании БД необходимо дать имя базе данных на латинском алфавите и выбрать способ хранения символов из набора *unicode* (рис. 2.1).

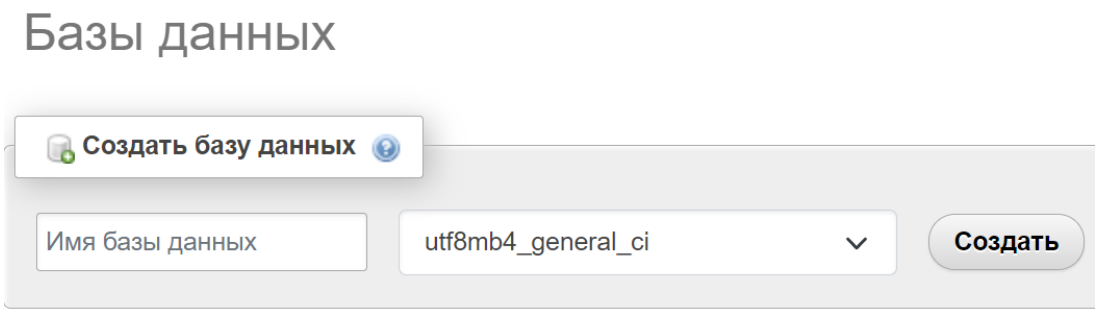


Рис. 2.1. Создание БД

Внутри базы связанные данные группируются в таблицы, каждая из которых состоит из строк и столбцов. Чтобы при проектировании модели

базы данных обеспечить согласованность разных записей, нужно назначить соответствующий тип данных для каждого столбца (рис. 2.2).

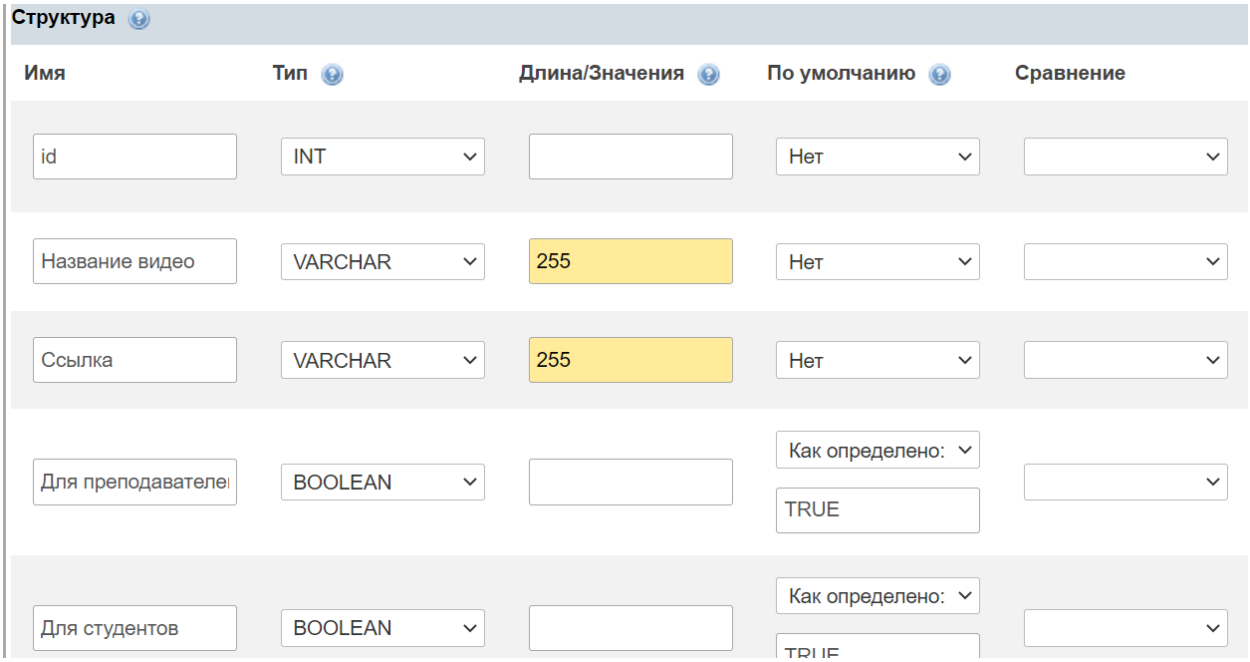


Рис. 2.2. Структура БД

Традиционно, язык программирования *PHP* поддерживает работу с такой базой данных, как *MySQL* (это СУБД, поддерживающая структурированный язык запросов *SQL*). Для работы с базой данных *MySQL* в *PHP* встроены специальные функции. При подключении к *MySQL* соответствующий сценарий исполняет запрос и показывает результат запроса.

Но прежде, чем выполнить какую-нибудь операцию с записями, находящимися в БД, нужно сначала к этой БД подключиться.

Выполнить подключение к серверу БД *MySQL* можно несколькими способами:

- с помощью объектно-ориентированного подхода *MySQLi*;
- с помощью процедурного подхода *MySQLi*;
- используя технологию *PDO*;
- используя способ подключения для старых версий *PHP*.

Но какой бы метод выбран не был, важно иметь основную информацию как о *database*, так и нюансах доступа (*access*) к ней. Что подразумевается под основной информацией:

- сервер размещения. Когда разработка проекта ведется на локальном сервере, указывается значение *localhost*;
- *login* и *password* пользователя, имеющего доступ (*access*). Когда работы проводятся на локальном сервере, юзер может быть *root* и с пустым паролем;
- имя базы данных.

Технология подключения подбирается программистом с учетом условий проекта, личных предпочтений, а также удобства применения.

Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованны в виде таблицы (рис. 2.3), состоящей из столбцов и строк. В таблице хранится информация об объектах, представленных в базе данных. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута. Каждая строка в таблице может быть помечена уникальным идентификатором, называемым первичным ключом.

id	Название видео	Ссылка	Для учителей	Для студентов
1	Вход на персональную страницу и настройка профиля	css/vid1.mp4	0	1
2	Просмотр доступных курсов и загрузка выполненного ...	css/vid2.mp4	0	1
3	Тестирование	css/vid3.mp4	0	1
4	Режим редактирования и загрузка файла на курс	css/vid4.mp4	1	0
5	Создание и правильная настройка элемента тест	css/vid5.mp4	1	0
6	Вопросы с множественным выбором ответов	css/vid6.mp4	1	0
7	Создание категории в банке вопросов	css/vid7.mp4	1	0

Рис. 2.3. Хранение данных в базе данных

В столбце «Для учителей» видео для преподавателей равно единице, аналогично столбцу «Для студентов», где видео для студентов равно единице, а для преподавателей равно нулю.

2.3. Описание пользовательских подпрограмм

При разработке программного продукта был составлен алгоритм, отображающий взаимосвязь основных блоков программы (Приложение 1). Программный продукт состоит из пользовательских подпрограмм, листинг которых представлен в Приложении 2.

Для того чтобы работать с БД, нужно получить доступ к ней, имея основную информацию, присвоенную набору переменных. Файл *dbconn.php* подключается к базе данных (рис. 2.4). Если есть ошибка в информации о базе данных, то выводится сообщение о невозможности подключения к базе данных.

```php
<?php
$pdo = NULL;
try{
    $pdo = new PDO("mysql:host=localhost;dbname=video_galereia","admin","S4w-NSM-Tmz-74f");
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $pdo->exec('SET NAMES "utf8"');
}
catch(PDOException $e){
    $errorMessage = "Невозможно подключиться к базе данных: ".$e->getMessage();
    die();
}
```

Рис. 2.4. Процедурный подход к применению *MySQLi*

Для соединения и взаимодействия с базой данных в *PHP* есть несколько функций:

- *mysqli_connect* – функция устанавливает соединение с сервером базы данных;
- *mysqli_query* – отправляет *SQL* запрос к базе данных;
- *mysqli_close* – закрывает соединение с базой данных.

Функция *mysqli_connect* принимает три параметра: адрес сервера (хост), имя пользователя, пароль пользователя. Функция *mysqli_connect* возвращает данные, описывающие соединение, которые в дальнейшем должны передаваться функции *mysqli_query*.

Все манипуляции с базой происходят с помощью различных *SQL* запросов через функцию *mysqli_query*. С помощью *SQL* запросов можно создавать и удалять таблицы, делать выборки данных по фильтру с различными сортировками, добавлять и удалять строки. Функция *mysqli_query* принимает два параметра: первый – это данные, описывающие соединение (результаты работы функции *mysqli_connect*), второй – это *SQL* запрос в виде простой строки.

Далее сформирована простая *html* форма для поиска (рис. 2.5).

```
<div class="vegukonem-nenkaepren">
    <div class="vedanageous">
        <form>
            <input class="search" type="text" name="search" placeholder="Искать...">
            <button class="btn"  type="submit"><img src="img/kr.png" alt=""
            width="40" class="picture" ></button>
        </form>
    </div>
</div>
</div>
```

Рис. 2.5. Форма для поиска: а – *html* код для создания поиска

Рис. 2.5. Форма для поиска: б – результат *html* кода

Для того чтобы извлекать дынные из базы данных по какому-то параметру был создан файл *get_videos.php* файл. Параметром является либо пустая строка, которая выводит все видео на странице, либо *$_GET["search"]*, *"search"* – это ключ, в котором хранятся данные. С

помощью *get_videos.php* пользователь сможет получить данные по запросу (рис. 2.6). Так как видео разделяются на видео для преподавателей и для студентов в *get_videos.php* есть две функции: функция *getVideos*() отвечает за получение данных по запросу на странице для преподавателей, а функция *getVideo*() отвечает за получение данных по запросу на странице для студентов. *require_once* – функция, которая подключает файл *dbconn.php*.

```php
<?php
require_once("dbconn.php");
function getVideos($pdo,string  $str="", $forTeachers = 1){
    $result = NULL;
    if($str=="") {
        $sql="SELECT `id`, `Название видео`, `Ссылка` FROM `video` WHERE `Для учителей`=:forTeachers";
        $result = $pdo->prepare($sql);
        $result->execute(array("forTeachers"=>$forTeachers));
    }
    else{
        $sql = "SELECT `id`, `Название видео`, `Ссылка` FROM `video` WHERE
        `Название видео`=:video_name AND `Для учителей`=:forTeachers";
        $result = $pdo->prepare($sql);
        $result->execute(array("video_name"=>$str,"forTeachers"=>$forTeachers));
    }
    return $result->fetchAll(PDO::FETCH_ASSOC);
}

function getVideo($pdo,string  $str="", $forTeachers = 1){
    $result = NULL;
    if($str=="") {
        $sql="SELECT `id`, `Название видео`, `Ссылка` FROM `video` WHERE `Для студентов`=:forTeachers";
        $result = $pdo->prepare($sql);
        $result->execute(array("forTeachers"=>$forTeachers));
    }
    else{
        $sql = "SELECT `id`, `Название видео`, `Ссылка` FROM `video` WHERE
        `Название видео`=:video_name AND `Для студентов`=:forTeachers";
        $result = $pdo->prepare($sql);
        $result->execute(array("video_name"=>$str,"forTeachers"=>$forTeachers));
    }
    return $result->fetchAll(PDO::FETCH_ASSOC);
}
```

Рис. 2.6. Извлечение дынных из базы данных

В файле *main.js* (рис. 2.7) выполняется привязка к событию и *ajax* запрос, его задача получить из строки поиска строку, которую вводит пользователь, потом эту строку с помощью *Get* запроса передать файлу *prep.php/ stud.php* (рис. 2.8), который позволит сгенерировать нужное видео. *console.log(data) – функция,* предназначенная для вывода результата в

консоль браузера. *Success* – задает функцию, которая будет вызываться в случае успешного выполнения запроса.

```
$(document).ready(function(){
    $(".btn").on("click", function(e){
        e.preventDefault();
        let form = $(this)[0].form;
        let data = $(form).serializeArray();
        $.ajax({
            type:"GET",
            url:window.location,
            data: data,
            success:function(data){
                console.log(data);
                /*$(".search")[0]["value"]="";
                for(let i=0; i<data["videos"]["length"];i++){
                    $(".container-2").append(data["template"]);
                }*/
                let container=$(data).find(".container");
                $(".container")[0].innerHTML=container[0].innerHTML;
            },
            error:function(data){
                console.log(data);
            }
        });
    })
```

Рис. 2.7. Файл *main.js*

```
<?php
require_once("includes/dbconn.php");
require_once("includes/get_videos.php");
$videos="";
if(isset($_GET["search"])){
  $_GET["search"] = htmlspecialchars($_GET["search"]);
  $videos = getVideos($pdo,$_GET["search"]);
}
else {
  $videos = getVideos($pdo,"",TRUE);
}
?>
```

Рис. 2.8.Файл *prep.php/ stud.php*

В файле *prep.php* и *stud.php* с помощью require_once подключается база данных и функции *getVideos* и *getVideo*. Функция *isset* будет возвращать значение *tru* или *false*.

2.4. Руководство пользователю

Для начала пользователь должен зайти на сайт «Видео*Pi*», на главную страницу (рис. 2.9).

Рис. 2.9. Главная страница сайта «Видео*Pi*»

Далее пользователь должен войти на страницу либо для преподавателей, либо для студентов, нажав на меню «Видео» и перейдя по ссылке (рис. 2.10).

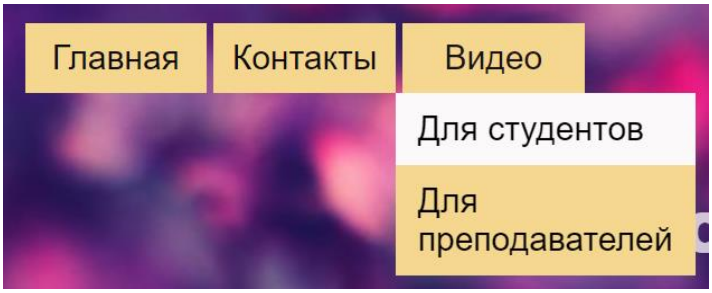


Рис. 2.10. Меню на главной странице

Далее пользователь попадает на страницу для студента или для преподавателя и, чтобы пользователю найти нужную информацию, нужно ввести в поле поиска в правом верхнем углу страницы запрос – точное название видео (рис. 2.11).

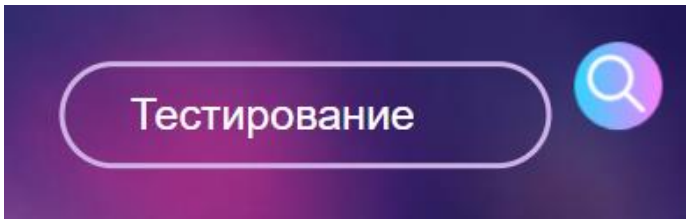


Рис. 2.11. Поле поиска на странице

Запрос в поисковике – это словосочетание, отдельные слова или комбинация знаков, которые вводит пользователь в строку поиска для того, чтобы узнать нужные данные в интернете. После введения в поле поиска запроса нужно кликнуть на кнопку поиска.

Далее на странице появится результат запроса пользователя – видео с нужной информацией (рис. 2.12).

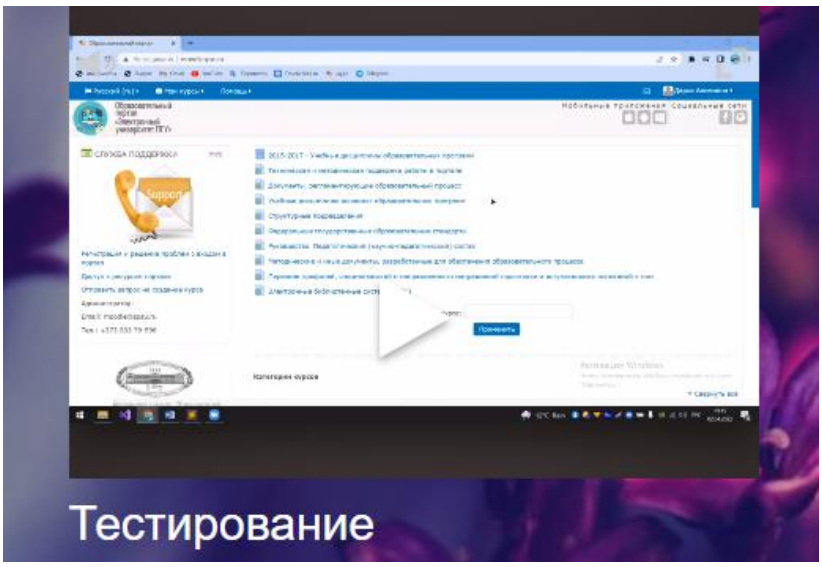


Рис. 2.12. Результат запроса пользователя

После просмотра нужной информации пользователь может сделать еще запросы в поисковике, пройти по ссылке на главной странице на предоставленные сайты moodle, ПГУ или РФПГУ или же выйти и закрыть страницу.

2.5. Тестирование и анализ полученных результатов

При создании информационно-поисковой системы, в самом начале, была выявлена проблема. При наличии всех теоретических знаний, не хватало практических для выбора метода написания поисковой системы и ее дальнейшего создания. После выбора метода написания программы, было выявлено, что из-за малых знаний о *php* и подключении базы данных, база данных не корректно подключалась к *php* и программа не отвечала на запросы пользователя.

Во время создания программного продукта возникли проблемы. Из-за *bootstrap* поиск сдвинулся вправо за пределы страницы. Исправили, добавив поиск в секцию с меню. При начале работы с БД возникла проблема с созданием пользователя в БД. Вскоре ошибка была решена. Также скрипт *get_videos.php* выдавал неправильно информацию – выводил код как строку.

Проблема возникла из-за того, что путь в *URL* был неправильно прописан. Также возникла проблема с файлом *main.js*, из-за того, что папка «курсовая», в которой находится этот файл, написана на русском, проблема была решена. Методы для удаления строки из поисковика *val* и *arr* в файле *main*.js не выполнялись свою работу из-за того, что код закэшировался. Чтобы решить эту проблему, в файле *index.php* добавлено скрипт *main.js* текущее время – *time*.

В ходе практической работы были выявлены сложности создания ИПС на *Visual Studio Code* и с созданием БД *в XAMPP* из-за малого количества знаний о функциях в *php* и работой с БД.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы на тему «Информационно-поисковые системы» поставленная цель была достигнута. При достижении поставленной цели были решены следующие задачи:

- исследовано понятие и виды информационно-поисковой системы;
- проанализирована структура информационно-поисковой системы;
- изучены элементы информационно-поисковой системы;
- рассмотрены виды существующих программных продуктов;
- выбрана среда программирования;
- создана структура базы данных;
- проведены тестирование программы и анализ полученных результатов.

В ходе выполнения курсовой работы была изучена новая область, а именно работа с базой данных, а также такие языки программирования, как *SQL* и *PHP* в сборке локального веб-сервера *XAMPP*.

В ходе тестирования программного продукта были выявлены следующие недостатки:

- поисковая система ищет только при полном соответствии запроса пользователя с названием видео;
- запрос в поле поиска можно вводить только на страницах для студентов и преподавателей.

Преимуществом является: подключение информационно-поисковой системы к образовательной платформе облегчает и помогает, ускоряет обучение работе с образовательным порталом «Электронного университета ПГУ».

В перспективе планируется сделать фильтрацию поиска, отрегулировать правила, по которым пользователь может вводить запросы и сделать возможным введение поиска на главной странице.

Подводя итоги, можно сказать, что в ходе создания программного продукта было получено много знаний о *html*, *php*, *MySQLi*. Данный программный продукт облегчит и ускорит обучение студентов и преподавателей работе с образовательным порталом «Электронного университета ПГУ» на сайте «Видео*Pi*».

СПИСОК ЛИТЕРАТУРЫ

1. Курсовое проектирование по основам программирования: Методические рекомендации. /Сост.: Е.С. Гарбузняк, Н.В. Чернега, О.М. Нагаевский. – Рыбница, 2015. – 115 с.

2. *Mediagnosis* [Электронный ресурс]. – Каптерев А.И. Электронный учебник по информатике. – Режим доступа: *http://www.mediagnosis.ru/Autorun/Page6/10_1_.htm.*

3. *Allbest* [Электронный ресурс]. – Информационно-поисковые системы. – Режим доступа: *https://knowledge.allbest.ru/programming/3c0a65625a3ac68b4c43b88521306c36_0.html.*

4. *Wikipedia* [Электронный ресурс]. – Релевантность. – Режим доступа: *https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D0%B5%D0%B2%D0%B0%D0%BD%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C.*

5. Бобродобро [Электронный ресурс]. – Информационно-поисковые системы. – Режим доступа*: https://prog.bobrodobro.ru/23629.*

6. *Domashnya* [Электронный ресурс]. – Информационно-поисковые системы. – Режим доступа: *https://tsput.ru/res/informat/sist_seti_fmo/lekcii/lekciy-4.html*

7. Инфоурок [Электронный ресурс]. – Информационно поисковые системы и их классификация. – Режим доступа: *https://infourok.ru/informacionno-poiskovye-sistemy-i-ih-klassifikaciya–5243207.html.*

8. *StudentLib* [Электронный ресурс]. – Информационно поисковые системы. – Режим доступа: *//studentlib.com/chitat/kursovaya_rabota_teoriya-74065-informacionno_poiskovye_sistemy.html.*

9. *Book-science* [Электронный ресурс]. – Структуры информационно-поисковых массивов в ИПС. – Режим доступа: *http://book-science.ru/exact/it/struktury-informacionno-poiskovyh-massivov-v-ips.html.*

10. *Studbooks.net* [Электронный ресурс]. – Элементы ИПС. – Режим доступа: *https://studbooks.net/722515/zhurnalistika/elementy.*

11. *It-black.ru* [Электронный ресурс]. – Обзор поисковых систем. – Режим доступа: *https://it-black.ru/obzor-poiskovyh-sistem/.*

12. Как Просто! [Электронный ресурс]. – Достоинства и недостатки поисковых систем. – Режим доступа: *https://www.kakprosto.ru/kak-819840-dostoinstva-i-nedostatki-poiskovyh-sistem#ixzz7SMLx4axE.*

13. *RGRU* [Электронный ресурс]. – "Яндекс" поздравил *Google* с 20-летием, перекрасив свой логотип. – Режим доступа: *https://rg.ru/2018/09/27/iandeks-pozdravil-google-s-20-letiem-perekrasiv-svoj-logotip.html.*

14. Налоги в Казахстане [Электронный ресурс]. – *Googl*e отобрала у *Apple* звание самого дорогого бренда. – Режим доступа: *https://nalogikz.kz/docs/google-otobrala-u-apple-zvanie-samogo-dorogogo-brenda.html.*

15. *BIZZAPPS* [Электронный ресурс]. – Почта *Mail.ru*. – Режим доступа: *https://bizzapps.ru/p/pochta-mail-ru/.*

16. *ADD – ONS* [Электронный ресурс]. – DuckDuckGo Privacy Essentials. – Режим доступа: *https://addons.mozilla.org/ru/firefox/addon/duckduckgo-for-firefox/.*

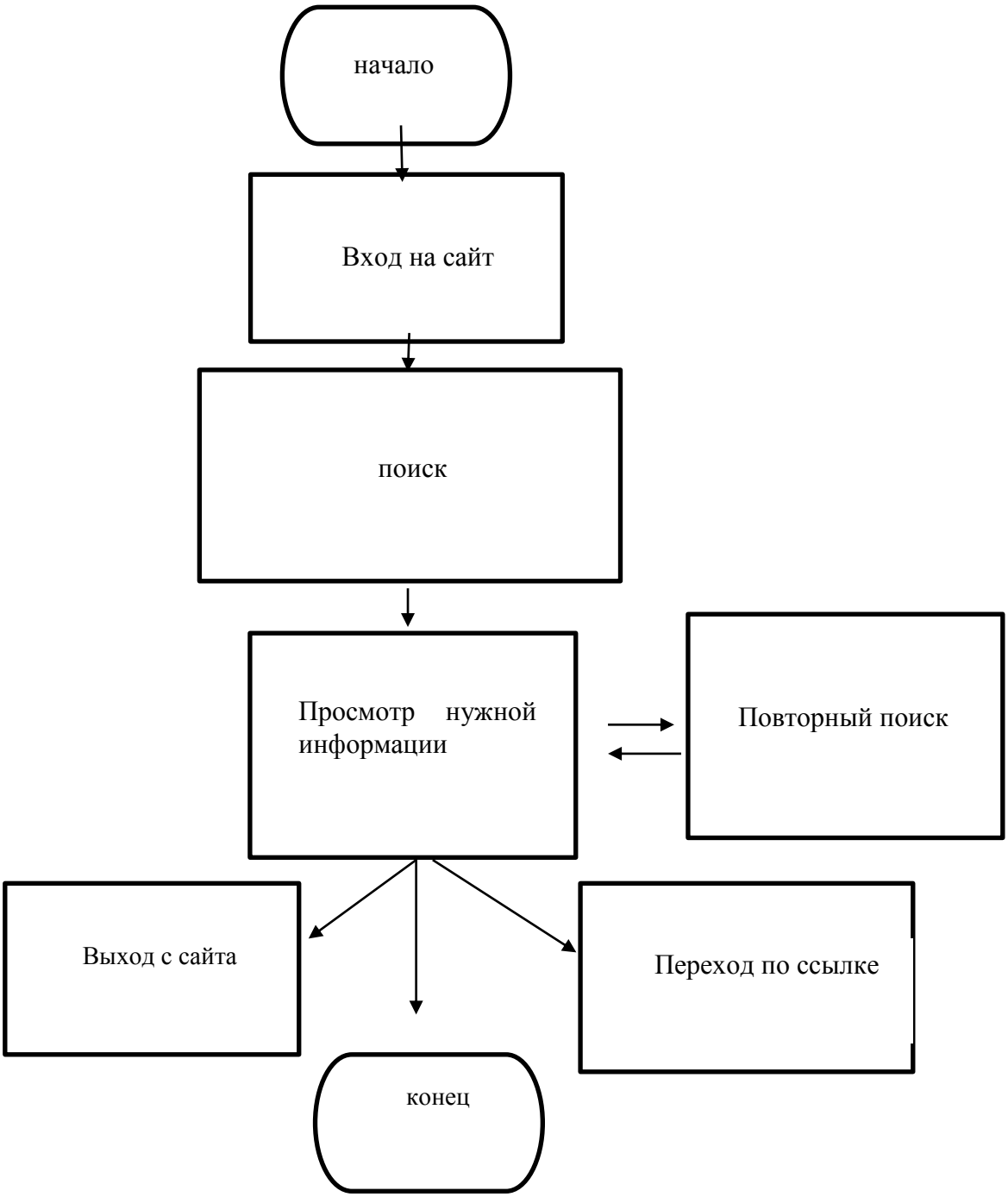
17. *Wikipedia* [Электронный ресурс]. – *Microsoft Visual Studio.* – Режим доступа: *https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio.*

18. *Otos Journal* [Электронный ресурс]. – Работа с базой данных в *PHP*. – Режим доступа: *https://otus.ru/journal/rabota-s-bazoj-dannyh-v-php/.*

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1

Алгоритм работы программного продукта



ПРИЛОЖЕНИЕ 2

Листинг программного продукта

```
//-------------------------index.php----------------------------//
<div>
<div class="vegukonem-nenkaepren">
<div class="vedanageous">
<form>
<input class="search" type="text" name="search" placeholder="Искать...">
<button class="btn"  type="submit"><img src="img/kr.png" alt="" width="40" class="picture" ></button>
<!--input-teg, class-atribut-->
</div>
</div>
</form>
</div>
//-----------------------------style.css-----------------------------//
.vegukonem-nenkaepren{/* начало поиска*/
    position: absolute;
    top: 10%;
    left: 80%;
    transform: translate(-90%,-60%);
  }

  .vedanageous{
    position: relative;
    padding: 20px 0px;
  }

  .search{
    width: 160px;
    border: 3px solid rgb(208, 178, 231);
    background: transparent;
    padding: 10px px;
    border-radius: 100px;
    outline: none;
    font-size: 20px;
    color: #ffffff;
  }

  ::-webkit-input-placeholder {
    color: #ffffff;
    font-family: Roboto;
```

```
      text-transform: uppercase;
    }
    ::-moz-placeholder {
      color: #a44f4f;
    }
    :-ms-search-placeholder {
      color: #eaeaea;
    }

    .kopaverage-dapodcas {
      position: absolute;
      top: 21px;
      right: 49px;
      cursor: pointer;
      color: #ecd27b;
      background: #eaeaea;
      border: 0px;
      width: 118px;
      height: 57px;
      border-radius: 100px;
      outline: none;
      text-transform: uppercase;
      font-weight: bold;
    }
    button{
       border: 0;
       outline: none;
       background: transparent;
       }

//-------------------------prep.php----------------------//
<?php
require_once("includes/dbconn.php");
require_once("includes/get_videos.php");
$videos="";
if(isset($_GET["search"])){
  $_GET["search"] = htmlspecialchars($_GET["search"]);
  $videos = getVideos($pdo,$_GET["search"]);
}
else {
  $videos = getVideos($pdo,"",TRUE);
}
?>
<div class="container">
```

```
 <?php foreach($videos as $video):?>
<iframe width="100%" height="300" style="border:none;"
src="<?=$video["Ссылка"]?>"
frameborder="0" allow="clipboard-write; encrypted-media; gyroscope; picture-in-
picture" allowfullscreen>
</iframe>
<p><?=$video["Название видео"]?></p>
<?php endforeach;?>
</div>
//-------------------prep.css/ stud.css--------------------//
.container{
   display: block;
   margin-left: auto;
   margin-right: auto
   }
//----------------------stud.php----------------------------//
<?php
require_once("includes/dbconn.php");
require_once("includes/get_videos.php");
$videos="";
if(isset($_GET["search"])){
  $_GET["search"] = htmlspecialchars($_GET["search"]);
  $videos = getVideo($pdo,$_GET["search"]);
}
else {
  $videos = getVideo($pdo,"",TRUE);
}
?>
<div class="container">
<?php foreach($videos as $video):?>
<iframe width="100%" height="300" style="border:none;"
src="<?=$video["Ссылка"]?>"
frameborder="0" allow="clipboard-write; encrypted-media; gyroscope; picture-in-
picture" allowfullscreen>
</iframe>
<p><?=$video["Название видео"]?></p>
<?php endforeach;?>
</div>
//--------------------dbconn.php-------------------------//
<?php
$pdo = NULL;
try{
```

```
   $pdo = new PDO("mysql:host=localhost;dbname=video_galereia","admin","S4w-NSM-Tmz-74f");
   $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
   $pdo->exec('SET NAMES "utf8"');
}
catch(PDOException $e){
   $errorMessage = "Невозможно подключиться к базе данных: ".$e->getMessage();
   die();
}
//----------------------get_videos.php--------------------//
<?php
function getVideos($pdo,string  $str="", $forTeachers = 1){
   $result = NULL;
   if($str=="") {
      $sql="SELECT `id`, `Название видео`, `Ссылка` FROM `video` WHERE `Для учителей`=:forTeachers";
      $result = $pdo->prepare($sql);
      $result->execute(array("forTeachers"=>$forTeachers));
   }
   else{
      $sql = "SELECT `id`, `Название видео`, `Ссылка` FROM `video` WHERE `Название видео`=:video_name AND `Для учителей`=:forTeachers";
      $result = $pdo->prepare($sql);
      $result>execute(array("video_name"=>$str,"forTeachers"=>$forTeachers));
   }
   return $result->fetchAll(PDO::FETCH_ASSOC);
}

function getVideo($pdo,string  $str="", $forTeachers = 1){
   $result = NULL;
   if($str=="") {
      $sql="SELECT `id`, `Название видео`, `Ссылка` FROM `video` WHERE `Для студентов`=:forTeachers";
      $result = $pdo->prepare($sql);
      $result->execute(array("forTeachers"=>$forTeachers));
   }
   else{
      $sql = "SELECT `id`, `Название видео`, `Ссылка` FROM `video` WHERE `Название видео`=:video_name AND `Для студентов`=:forTeachers";
      $result = $pdo->prepare($sql);
      $result->execute(array("video_name"=>$str,"forTeachers"=>$forTeachers));
```

```
    }
    return $result->fetchAll(PDO::FETCH_ASSOC);
}
//------------------------ main.js--------------------//
$(document).ready(function(){
$(".btn").on("click", function(e){
e.preventDefault();
let form = $(this)[0].form;
let data = $(form).serializeArray();
$.ajax({
type:"GET",
url:window.location,
data: data,
success:function(data){
.log(data);
/*$(".search")[0]["value"]="";
for(let i=0; i<data["videos"]["length"];i++){
$(".container-2").append(data["template"]);
}*/
let container=$(data).find(".container");
$(".container")[0].innerHTML=container[0].innerHTML;
},
error:function(data){
console.log(data);
}
});
})
});
```