МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА ДИНАМИЧЕСКОГО САЙТА С ИСПОЛЬЗОВАНИЕМ PHP И MYSQL

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Целуйкиной Светланы Сергеевны

Научный руководитель
доцент, к. п. н. _____ В. А. Векслер

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2021

ВВЕДЕНИЕ

Актуальность темы. В современном мире веб-сайты используются повсеместно для отображения информации по некоторой тематике. Сайт может являться энциклопедией животных, а может быть интернет-магазином одежды. Практически каждая организация имеет собственный сайт, чтобы люди могли найти информацию о предоставляемых товарах и услугах этой компании при помощи Интернета. Мир каждый день меняется, все больше цифровизации внедряется в жизнь человека, а значит, разработка веб-сайтов является очень актуальной темой на данный момент времени. Для создания сайта существуют разные языки программирования, однако все они реализуют некую основу, универсальную для каждого из них. Для разработки серверной стороны сайта удобно использовать язык PHP, так как он достаточно актуален и широко применяется на данный момент времени: на нем успешно можно реализовать почти все необходимые функциональности, а также программы, написанные на нем, являются защищенными, благодаря закрытому исходному коду. Для хранения большого количества информации следует использовать реляционные базы данных, одним из представителей которых является MySQL.

Цель бакалаврской работы – разработать динамический сайт с использованием PHP и MySQL.

Поставленная цель определила **следующие задачи**: изучение подходов разработки веб-сайта, ознакомление с научной литературой по заданной проблематике, разработка собственного сайта с использованием PHP и MySQL.

Методологические основы разработки динамического сайта с использованием PHP и MySQL представлены в работах Р. Никсона, Д. Скляра, М. Зандстры, Э. Фримена, Д. Дакетта, П. Дюбуа, Л. Ульмана, К. Дугласа, К. Уодтке.

Практическая значимость бакалаврской работы заключается в следующем: данную работу можно использовать в учебных целях при изучении структуры веб-сайта; для разработки собственного сайта; а также веб-сайт, приведенный в текущей работе, может быть полезен учащимся в художественных школах, которым необходимо получить информацию о художниках и скульпторах Саратовской области.

Структура и объём работы. Бакалаврская работа состоит из введения, трех разделов, заключения, списка использованных источников, одного при-

ложения. Общий объём работы — 89 страниц, из них 56 страниц — основное содержание, включая 23 рисунка, цифровой носитель в качестве приложения, список использованных источников информации — 25 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Сущность веб-сайта» посвящен структуре сайта. Сюда включены необходимые определения и понятия:

- интернет — глобальная система соединенных компьютерных сетей, которая используется для передачи различной информации;
- доменное имя представляет собой уникальное имя сайта, позволяющее отделять его от других сайтов сети Интернет;
- веб-сайт — это некоторый адрес, находящийся в сети Интернет, на котором расположена информация по определенной тематике;
- сервер (или же хостинг) — компьютеры, хранящие файлы ресурсов.

Любой сайт построен по принципу «клиент-сервер» — сетевая архитектура, в которой есть заказчики услуг, называющиеся клиентами, и поставщики услуг, которые называются серверами. Клиент посылает некоторый запрос, сервер его обрабатывает (возможно, с использованием информации, хранящийся в базе данных) и возвращает ответ клиенту. Использование протокола HTTP удобно в данной архитектуре.

Существуют несколько основных HTTP-методов, чтобы разграничить работу с ресурсами:

- GET — получение информации с сайта;
- POST — создание нового ресурса на сайте;
- PUT — обновление текущих данных;
- DELETE — удаление информации с сайта.

Очень важным атрибутом ответа сервера является код состояния. Он показывает результат запроса, на основании которого пользователь предпринимает свои дальнейшие решения. Существуют пять классов состояния:

- Информационные (1xx) — предназначены для указания процесс передачи. В версии HTTP 1.1 клиент распознает этот класс сообщений как обычный ответ и ничего серверу не отправляет.
- Успех (2xx) — показывают, что запрос пользователя был принят и обработан успешно. Сервер может также отправить в ответ заголовки и тело сообщения с зависимости от статуса.

— Перенаправление (3xx) — показывают необходимость выполнения запроса по другому URI для успешной обработки. Адрес перенаправления указывается сервером в заголовке Location.
— Ошибка клиента (4xx) — необходимы, чтобы обозначить ошибки на стороне клиента. В теле сообщения сервер возвращает гипертекстовое пояснение ошибки для любого метода, кроме HEAD.
— Ошибка сервера (5xx) — предназначены для необработанных исключений, когда происходит обработка какой-либо операции со стороны сервера. Как и в случае с ошибкой клиента, в ответ сервер возвращает гипертекстовое объяснение ошибки в теле сообщения при работе со всем методами, кроме HEAD.

У протокола HTTP есть существенный недостаток — он не защищен. Данные клиента передаются через другие компьютеры локальной сети, коммутаторы и маршрутизаторы, поэтому если злоумышленник получит доступ хотя бы к одному устройству, может произойти утечка данных. В HTTP не используется шифрование по умолчанию из-за следующих особенностей при шифровании: необходимо больше вычислительных мощностей, возрастает передача данных, запрет на использование кэширования. Но для передачи важной личной информации, например, паролей, необходимо произвести защиту от утечки, что реализовано в протоколе HTTPS.

Со стороны сервера может быть произведена различная обработка запросов клиента в зависимости от того или иного аспекта. Сервис может быть сетевым, получать задачи из базы данных или очереди задач. Существуют несколько различных подходов к реализации работы с запросами:

— последовательная обработка;
— процесс на запрос, поток на запрос;
— поток на запрос, пулл процессов или потоков;
— событийно-ориентированная обработка;
— полусинхронный паттерн;
— конвейерная обработка.

У каждого подхода есть свои преимущества и недостатки, например, самым простым подходом является последовательная обработка, однако здесь стоит серьезное ограничение на масштабирование, а конвейерная обработка обеспечивает высокую производительность, но довольно сложна в реализации

и не является пригодной для любого запроса. В итоге выбор той или иной архитектуры зависит от решаемой задачи.

Существует два типа сайтов: статические и динамические. В статическом сайте на каждый запрос какого-либо ресурса возвращается статический ответ сервера. Например, когда приходит GET запрос, сервер ищет требуемый файл в своей системе и возвращает ответ с помощью HTTP протокола, в котором передается найденный документ и успешный статус. В случае невозможности выдать данный файл, отправляется статус ошибки. Динамический сайт отличается от статического тем, что некоторая часть ответа создается динамически, когда возникает такая потребность. В данном типа сайта создание HTML-страниц происходит при помощи выборки данных из базы, которые вставляются в HTML-шаблоны. Данные в базе данных хранить удобнее, чем содержать их непосредственно в HTML- странице.

Передача сообщений может происходить при помощи языков, использующих теговую разметку. Тег — это некоторый элемент, представляющий собой служебное слово, которое заключено в угловые скобки. Текст, который необходимо передать, должен быть вставлен между тегами. Самыми распространенные теговыми языками являются XML (расширяемый язык разметки) и HTML (язык гипертекстовой разметки).

В XML при принятии какого-либо запроса сперва проводится его проверка на корректное соответствие по схеме. В случае прохождения проверки запрос считается правильным, значит, можно запустить алгоритм обработки запросов. Схема также позволяет быстрее разобраться с работой приложения людям, которые проводят разработку и тестирование. Разработчик сможет понять, в каком виде нужно составлять запросы, а тестировщик — в каком виде происходит формирование запроса. Схема, в отличие от технического задание, обновляется каждый раз при обновлении кода, а значит, не устареет со временем.

Язык разметки HTML представляет собой стандартизированный язык, который применяется для составления форматированного текста, интерпретирующийся браузером. В результате можно видеть на экране элементы вебстраницы, которые были заложены в HTML-документе. В HTML, как и в XML, используются теги, между которыми заключается содержимое элемента. Сам язык следует правилам, содержащимся в XML-файле объявления типа

документа. Этот файл описывает, какие теги, атрибуты и значения можно использовать в данной версии HTML, отсюда же следует, что для каждой версии создается свой файл с таким типом.

Таким образом, в первом разделе были рассмотрены основные определения и понятия, структура веб-сайта, передача данных при помощи HTTP-протокола, а также взаимодействие с запросами со стороны клиента и сервера.

Второй раздел «Использование Symfony и MySQL» посвящен технологиям, которые можно выбрать при разработке веб-сайта.

Для обработки запросов можно использовать Symfony — фреймворк, который написан на языке PHP. С его помощью можно быстро и удобно разрабатывать и поддерживать любые пользовательские веб-сайты. При создании новой страницы необходимо указать её адрес. Для этого в Symfony есть аннотация Route. Также необходимо сделать контроллер — функцию PHP, которая пишется разработчиком для создания страницы. На вход принимает запрос, в котором содержится информация, и использует его для создания ответа (в Symfony для этого применяется объект класса Response), который может содержать HTML контент, строку формата JSON или даже бинарный файл, например, изображение или PDF. Если в ответ контролер отправляет HTML-страницу, то лучше всего сделать визуализацию в виде некоторого шаблона. Для этого в Symfony создан движок с названием Twig: он представляет собой язык шаблонов, который является минимальным, мощным и довольно удобным в разработке.

В проекте располагаются несколько основных директорий, с которыми необходимо ознакомиться, прежде чем начать разрабатывать серверную часть веб-сайта. В них строится вся главная структура, которая организует работу всего сервера:

- директория `config/` содержит в себе все необходимые настройки. Здесь конфигурируются сервисы, пакеты и маршруты;
- в директории `src/` хранится весь PHP код;
- директория `templates/` содержит весь шаблоны `Twig`;
- директория `bin/` включается в себя главную команду `bin/console`, а также некоторые менее важные исполняемые файлы;
- в директории `var/` хранятся автоматически созданные файлы, например, кэш-файлы (`var/cache/`) или лог-файлы (`var/log/`);

— директория `vendor/` содержит в себе библиотеки сторонних приложений, которые были загружены при помощи менеджера `Composer`;
— директория `public/` представляет собой корневой документ проекта, в котором должны содержаться любые файлы с публичным доступом.

Для хранения данных можно использовать MySQL — скоростной, многопоточный, многопользовательский и мощный сервер баз данных, основанный на структурированном языке запросов. Базы данных в MySQL являются реляционными, то есть они хранят информацию в отдельных таблицах, а не помещают ее в одно большое хранилище. Структуры баз данных организованы в файлы, которые оптимизированы по скорости. MySQL является программным обеспечением с открытым исходным кодом. Это означает, что любой человек имеет доступ к использованию и модификации программного обеспечения. MySQL сервер очень быстр, надежен, мастабируем и прост в использовании. Он может быть запущен на компьютере или ноутбуке вместе с другими приложениями, веб-серверами, не запрашивая при этом особых требований. MySQL работает в клиент-серверных или встроенных системах. Система баз данных состоит из мультипоточных серверов SQl, которые поддерживают различные клиентские программы и библиотеки, административные инструменты и большой диапазон различных программных интерфейсов.

Таким образом, во втором разделе были рассмотрены структуры фреймворка Symfony и MySQL, выделены основные их преимущества и показаны необходимые функциональности для разработки сайта.

Третий раздел «Разработка веб-сайта» посвящен реализации динамического сайта с использованием Symfony и MySQL на тему «Художники и скульпторы Саратова и Саратовской области». Для работы приложения необходимо установить PHP версии 7.4.3, Symfony версии не ниже 4.22.0, а также MySQL версии 8.0.23. Для запуска приложения необходимо зайти в директорию `\artists\public` и в терминале написать `symfony server:start`, затем перейти по появившейся ссылке на сайт. Также нужно в корневой директории `artists` написать команду `php bin/console sockets:start-chat` для запуска сервера внутреннего чата, находящегося на сайте.

На главной странице сайта, если пользователь не авторизован, в центре отображается галерея некоторых картин (замкнутая циклически), слева — меню, справа — кнопки входа и регистрации.

Обработка со стороны сервера происходит при помощи контроллера в методе `bannerAction`. Функция принимает запрос и сессию как параметры, затем в базе данных при помощи репозитория `ImageRepository` ищутся все изображения, у которых флаг `slide` принимает истинное значение. Далее рассматривается текущая сессия, если она существует, то с помощью репозитория `UserRepository` ищется пользователь по соответствующему логину, и затем все данные передаются в HTML с помощью метода `render()`. Если же пользователь не авторизован, то в метод `render()` передается только массив полученных изображений.

При нажатии на кнопку `Войти` открывается страница авторизации, где необходимо ввести логин и пароль, что войти в учетную запись. Если пользователь еще не зарегистрирован, то он может пройти по ссылке регистрации. Также существует возможность возврата на главную страницу, если пользователь передумал делать вход в учетную запись.

При успешном входе в учетную запись пользователь попадет снова на главную страницу, но теперь справа будут отображаться его логин и кнопка `Выйти`, при нажатии на которую клиент выйдет из своей учетной записи и попадет на главную страницу сайта, которая была описана ранее. Также для зарегистрированных пользователей в меню появляются две новые возможности: пройти тест и написать в службу поддержки, нажав на кнопку `Помощь`. Если вход в учетную запись не удался, пользователю будет предложено снова ввести логин и пароль.

Существует также разграничение пользователей: обычный клиент, редактор и администратор. Редактору, помимо указанных до этого возможностей при успешной авторизации, предоставляется еще кнопка `Редактирование` на главной странице, и он может отвечать на все вопросы, которые задают обычные пользователи. У администратора появляется дополнительно кнопка `Администрирование` на главной странице сайта.

На странице регистрации пользователю предлагается придумать логин и пароль и нажать на кнопку `Зарегистрироваться`. Если логин уже существует, ему будет предложено придумать какой-либо другой. Также пользователь может вернуться на главную страницу, если передумает регистрироваться. При успешной регистрации клиент перейдет на страницу логина, где ему необходимо ввести свои логин и пароль для входа в учетную запись. На стороне

сервера при регистрации происходит занесение полученной информации в базу данных. Пароль заносится не в открытом виде, а при помощи шифрования md5.

Нажав на самую первую кнопку меню, пользователь будет переведен на страницу художников Саратовской области. В центре выделены ссылки на художников 19 и 20 веков, каждая из них ведет на соответствующую статью о нём. Справа на странице сделана кнопка `Назад`, которая ведет на главную страницу сайта. Обработка информации данной странице представлена в методе `artistsAction()` контроллера `ArtistsController`.

При нажатии на ссылку какого-либо художника пользователь перейдет на страницу, содержащую статью об этом художнике. По центру страницы будет описана основная информация о главных моментах жизни данного художника, вставлены некоторые значимые его картины, сверху указан его портрет, а также имя, отчество и фамилия. Справа сверху будет содержаться кнопка `Назад`, ведущая на предыдущую страницу с художниками. На текущей странице пользователь может узнать факты жизни интересующего его творческого человека. Статьи хранятся в базе данных, при обработке запроса в контроллере происходит с использованием репозитория выборка из базы информации по необходимому художнику и отображение её с помощью HTML.

В меню главной страницы также есть кнопка `Картины`, при нажатии на которую пользователь перейдет на страницу, где ему предложат выбрать одного из художников Саратовской области, чьи картины ему в данный момент интересны. Каждый художник изображен на портрете с соответствующей подписью, вверху справа также присутствует кнопка `Назад`, ведущая на главную страницу. После того, как пользователь выберет определенного художника, он попадет на страницу с картинами данного мастера. Каждая картина имеет подпись, справа в углу есть кнопка `Назад`, ведущая на страницу с художниками.

Обработка запроса со стороны сервера происходит следующим образом: для картин каждого художника представлена своя функция контроллера `ImagesController`, в которой происходит обращение к базе данных в таблицу `ArtistImage` за всеми картинами определенного художника, а затем отображение картин и подписи к ним на странице при помощи метода `render()`. Сами картины хранятся в отдельных папках для каждого художника, в базе данных располагаются только пути к файлу. Поиск производится при помо-

щи метода `findImagesByArtist()`, где выполняется запрос, представляющий собой выборку из таблиц художников, картин, а также совмещенную таблицу изображений и художников. Выборка производится по заданному художнику, результатом является массив, в котором содержится название картины и ее расположение.

Также пользователь может ознакомиться с информацией о скульпторах Саратовской области. Для этого ему нужно выбрать на главной странице в меню вкладку `Скульпторы`, после чего он попадет на страницу, где представлены четыре скульптора. Их портреты с подписями расположены по бокам страницы, в центре представлены ссылки, нажав на которые пользователь перейдет на статью о данном скульпторе.

После того, как пользователь выберет определенного скульптора и перейдет на страницу, он увидит статью, которая содержит основные моменты его биографии, а также несколько главных скульптур данного скульптора. Текст расположен по центру страницы, вверху представлены имя, отчество и фамилия этого скульптора, а также его портрет. Справа расположена кнопка `Назад` для возврата на предыдущую страницу со скульпторами. Статьи про скульпторов, как и статьи про художников, хранятся в базе данных и выбираются при помощи репозитория, обработка со стороны сервера проводится в контроллере.

Если пользователь захочет ознакомиться со скульптурами, то ему необходимо на главной странице сайта выбрать вкладку `Скульптуры` в меню. Он перейдет на страницу, где будет предложено выбрать одного из скульпторов, с чьими скульптурами он хотел бы ознакомиться. На данной странице изображены портреты скульпторов с подписями, справа сверху расположена кнопка `Назад`, ведущая на главную страницу.

Со стороны сервера для каждого скульптора происходит выборка его портретов в базе данных с помощью метода `findItems()`. В нем ищутся скульптор и соответствующий ему портрет в таблицах скульптора, скульптуры и таблицы, объединяющей скульпторов и скульптуры.

После выбора скульптора пользователь попадет на страницу со скульптурами. Скульптуры представлены в виде изображения, в верхем правом углу также присутствует кнопка `Назад`, которая ведет на предыдущую страницу. Обработка GET-запроса происходит в контроллере `SculptureController` с

помощью методов, предназначенных для каждого скульптора. Как и с картинами художников, в данном контроллере происходит запрос в базу данных при помощи репозитория для выборки всех скульптур конкретного скульптора, а затем происходит их отображение с подписью при помощи метода `render()`.

На текущем сайте существует возможность у пользователей высказывать свое мнение, делиться информацией и помогать другим посетителям при помощи форума. Незарегистрированные клиенты могут только читать сообщения других людей, в то время как у зарегистрированных пользователей появляется возможность добавить интересующую их тему и все вопросы, связанные с ней. Нажав на кнопку `Форум` на главной странице, пользователь перейдет на страницу с темами. Если он до этого вошел в свою учетную запись, то ему также будут представлены возможности добавления собственной темы. После создания темы пользователю необходимо перейти на появившейся ссылке для отправки сообщений на форум. Все отправленные сообщения сохраняются в базе данных и видны всем пользователям, включая тех, кто не вошел в учетную запись.

После создания темы пользователю необходимо перейти на появившейся ссылке для отправки сообщений на форум. Адрес каждой темы будет таким же, что и адрес форума, за тем исключением, что в адресной строке темы добавляется ее идентификатор. Так как тема только что была создана, то первоначально на странице будет предложение: «Пока никто не оставил сообщения здесь, будьте первым!». Также будет предложена форма для отправки сообщений и сама кнопка `Отправить`. После того, как пользователь отправит сообщение, на странице будет показан логин отправителя и информация, которую он написал, при этом поле отображаемых сообщений является недопустимым для записи. Все отправленные сообщения сохраняются в базе данных и видны всем пользователям, включая тех, кто не вошел в учетную запись.

Чтобы проверить, насколько хорошо пользователь смог изучить материал, представленный на данном сайте, ему предлагается пройти небольшой тест. Сперва необходимо войти в учетную запись, далее слева в меню выбрать вкладку `Пройти тест!` и нажать на неё. Если данный пользователь уже проходил тест, ему отобразится страница с таким сообщением и предложение пройти тест заново с учетом того, что старый результат будет перезаписан. Если пользователь нажмет на кнопку `Пройти заново` или же проходит тест

впервые, то в появившейся странице будет приглашение на прохождение теста и кнопка Начать. Затем на каждый странице будет задан один вопрос, представлено поле ответа и кнопка Следующий вопрос, при нажатии на которую сохранятся ответ на текущий вопрос. После ответа на последний вопрос пользователь попадет на страницу с результатом, и если было дано $2/3$ правильных ответов на вопросы, то тест считается пройденным успешно, иначе — неуспешно, и пользователю предлагается снова изучить материал на сайте. Пройти тест заново можно в любое время, результат последнего прохождения записывается в базу данных.

Если у пользователя возникает вопрос, он может обратиться с ним непосредственно к редакторам сайта. Для этого первоначально ему нужно зайти в свою учетную запись, а затем на главной странице выбрать вкладку Помощь. Пользователю откроется страница с чатом, вверху будет написано: «Приватный чат имя_пользователя с support», где support — любой редактор сайта (или же администратор). Далее будет представлено поле для ввода сообщения и кнопка Отправить, справа сверху есть кнопка На главную для возврата на главную страницу сайта. Передача сообщений происходит с помощью сокетов и отображается на странице сразу же, поэтому отправленное сообщение придет собеседнику незамедлительно, ему не нужно будет обновлять страницу, чтобы увидеть его.

Работа сокетов организована при помощи языка JavaScript на клиентской стороне в виде трех функций: `onopen()` — здесь устанавливается соединение и происходит вывод в консоль о том, что соединение установлено, `onmessage()` — в данной функции происходит отображение отправленных и полученных сообщений при помощи метода `appendMessage()`, а также функция `onerror()`, которая позволяет ловить и показывать сообщения, если чат начинает работать некорректно. Также отправка сообщений происходит при помощи функции `sendMessage(text)`, где параметр `text` — то, что отправляет пользователь. Сообщение отправляется в формате JSON, помимо него посылается информация о том, кто его отправил.

На стороне сервера сообщения обрабатываются при помощи четырех функций: `onOpen()`, `onMessage()`, `onClose()` и `onError()`. Функция `onOpen()` принимает в качестве аргумента соединение, прикрепляет его к текущему клиенту и выводит сообщение об этом. В методе `onMessage()`, который прини-

мает два параметра: сообщение и отправителя, — происходит декодирование информации из формата JSON, выполняется проверка, что отправитель не равен получателю, и если она успешна, происходит отправка сообщения. В функции `onClose()` соединение открепляется от текущего клиента и выводится информация об этом в консоль, а в методе `onError()` выводится сообщение об ошибке в консоль и закрывается соединение.

Все сообщения выводятся слева в виде «имя_пользователя: сообщение». Имя отправителя отображается как «me», имя собеседника — его логин, причем для обычного пользователя собеседник всегда отображается как «support».

Если клиент заходит в учетную запись как редактор, при переходе на страницу с чатом ему будут представлены в виде списка все пользователи, которые задавали вопрос по данному сайту. Помимо этого, на главной странице ему предоставляется кнопка `Редактирование`. Здесь возможно изменение картин, которые представлены в качестве баннера на главной странице. Все изображения отображаются в виде списка с подписями к ним, фразой «Добавить на главный экран» и полем с отметкой. Редактор следует выбрать, какие изображения он хочет выставить на главный экран, проставить галочки в полях с выбранными картинами и затем нажать на кнопку `Изменить`. Когда редактор вернется на главный экран, он сможет увидеть все выбранные ранее сообщения с помощью прокрутки баннера.

Администратор, как и редактор, тоже может заниматься редактированием или отвечать пользователям в качестве поддержки, однако, помимо этих функций, у него есть еще и возможность администрирования. В виде таблицы ему доступны идентификаторы пользователей, их логин, активность и роль, а также три кнопки: `Изменить`, `Отметить неактивным` и `Отметить активным`. В столбце, предназначенной для выбора роли пользователя, отображается текущая роль и в виде выпадающего списка предлагается сменить на другую. Также администратор может изменить активность пользователя, отметив его, например, неактивным с помощью соответствующей кнопки, если заметит подозрительную активность данного пользователя на сайте. Такая отметка делает автоматический выход из учетной записи отмеченного пользователя и не позволяет зайти в нее снова, пока администратор на сделает его активным при помощи одноименной кнопки.

ЗАКЛЮЧЕНИЕ

В результате изучения был разработан динамический сайт с использованием PHP и MySQL, а также решены остальные поставленные задачи: рассмотрены подходы разработки веб-сайта и изучена научная литература по заданной проблематике. Данное исследование может быть использовано для ознакомления с основными принципами работы веб-сайта, а также для разработки собственного динамического сайта с использованием PHP и MySQL. Разработанный сайт может быть использован учащимися художественной школы для изучения информации о художниках и скульпторах Саратовской области.

Отдельные части бакалаврской работы были опубликованы на конференции: ИТО 2021 «Современные информационные технологии в образовании».

Основные источники информации:

1. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р.Никсон. — СПБ.: Питер, 2019. — 816 с.
2. Скляр, Д. Изучаем PHP 7 / Д.Скляр. — СПБ.: Вильямс, 2017. — 464 с.
3. Фримен, Э. Изучаем программирование на JavaScript / Э.Фримен, Э.Робсон. — СПБ.: Питер, 2018. — 640 с.
4. Зандстра, М. PHP. Объекты, шаблоны и методики программирования / М.Зандстра. — СПБ.: Диалектика-Вильямс, 2019. — 736 с.
5. Скляр, Д. PHP. Рецепты программирования / Д.Скляр. — СПБ.: Вильямс, 2017. — 784 с.
6. Дакетт, Д. HTML и CSS. Разработка и дизайн веб-сайтов / Д.Дакетт. — М.: Эксмо, 2020. — 480 с.
7. Дюбуа, П. MySQL. Сборник рецептов / П.Дюбуа. — Спб.: Символ-Плюс, 2019. — 1056 с.
8. Ульман, Л. MySQL: Руководство по изучению языка / Л.Ульман. — М.: ДМК Пресс, 2019. — 354 с.
9. Дуглас, К. Как устроен JavaScript / К.Дуглас. — Спб.: Питер, 2019. — 304 с.
10. Уодтке, К. Информационная архитектура. Чертежи для сайта / К. Уодтке. — М.: Кудиц-Образ, 2019. — 320 с.