Министерство образования и науки РФ

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра Информационные технологии и автоматизированные системы

Базы данных

Лабораторная работа № 5

Тема: «OpenServer и MySQL»

Выполнил: студент группы Миннахметов Э.Ю.

Проверил: доцент кафедры ИТАС

Петренко А.А

г. Пермь – 2020

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

1 АНАЛИЗ ПОСТАВЛЕННОЙ ЗАДАЧИ 4

2 ТЕХНОЛОГИЯ РЕАЛИЗАЦИИ 5

2.1 Архитектура приложения 5

2.2 Ключевые JavaScript-функции 6

2.3 Ключевые PHP-функции 6

2.4 Новые SQL-запросы 9

ЗАКЛЮЧЕНИЕ 11

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 12

# ВВЕДЕНИЕ

Цель: разработать веб-интерфейс для работы с СУБД MySQL.

Задачи:

1. изучить подобный веб-интерфейс (phpMyAdmin);

2. описать структуру приложения;

3. разработать приложение с помощью php или любого другого языка программирования.

# 1 АНАЛИЗ ПОСТАВЛЕННОЙ ЗАДАЧИ

OpenServer – набор, который содержит в себе многие популярные инструменты веб-разработки, среди которых локальный сервер Apache, язык серверного программирования PHP (Hypertext Preprocessor – препроцессор гипертекста), СУБД MySQL, а также веб-клиент данной СУБД phpMyAdmin.

Серверные приложения можно писать на многих языках программирования, например, C#, Java, Python, Perl и т.д., но выбор был поставлен на языке PHP, т.к. он обладает низким уровнем вхождения и имеется в наборе веб-разработки OpenServer.

Для данной лабораторной работы HTML и CSS для фронт-энда будет недостаточно, поскольку необходимо будет предусмотреть динамическое добавление полей при создании таблицы базы данных. Эту роль возьмет на себя JavaScript, как язык, доказавший всему миру привлекательности динамических веб-страниц.

В качестве учебного проекта нет необходимости реализовывать полноценный веб-интерфейс для работы с СУБД MySQL, поскольку прежде всего нужно автоматизировать ранее полученные навыки, среди которых запросы на выборку данных о таблица и о содержимом в них, которые в данной работе будут генерироваться кодом на языке PHP, генерируемые кодом запросы на добавление записей в таблицы, запросы на создание таблицы базы данных, а также на их удаление.

**Вывод**, бэк-энд веб-приложения будет написан на связке языков PHP + MySQL, а фронт-энд на HTML + CSS + JavaScript.

**2 ТЕХНОЛОГИЯ РЕАЛИЗАЦИИ**

Разработка любого приложения начинается с построения его архитектуры, веб-приложение данной лабораторной работы не будет являться исключением, после чего будут рассмотрены ключевые функции PHP- и JavaScript-программ, а также SQL-запросы, используемые в веб-приложении.

**2.1 Архитектура приложения**

Особых изысков данного приложение требовать не будет и можно вообще обойтись без ООП, довольствуясь лишь парадигмой структурного программирования. Однако и такой код следует строить по архитектурному паттерну ***MVC*** (Model-View-Controller, Модель-Представление-Контроллер).

В качестве ***Модели*** будет взята ранее построенная таблица базы данных, используемая во второй и третьей лабораторной работе. За работу с моделью будут отвечать функции из файла ***database.php***.

За ***Представление*** будет отвечать файл ***layout.php***, содержащий в себе основу для вывода ***html***-кода, ***tableView.php*** – шаблон для вывода таблицы, ***createTable.php*** – страница создания таблицы базы данных, ***script.js*** – весь ***JavaScript***-код программы для динамичного создания новых полей в форма создания таблицы, и файл ***style.css***, отвечающий за стили элементов ***html***-кода.

***Контроллером*** в данном приложении будет выступать файл ***index.php*** – он будет обрабатывать запросы и запускать вывод ***html***-кода из функций подключенных ***php***-файлов.

**Подводя итог**, следует сказать, что архитектура веб-приложения была построена и теперь необходимо перейти к детальному рассмотрению SQL-запросов.

**2.2 Ключевые *JavaScript*-функции**

JavaScript-код стал необходим данному приложению после постановки требования динамически добавлять новые поля в HTML-страницу создания новой таблицы базы данных. Нижу приведен данный код.

Листинг 2.1 – JavaScript-код веб-приложения



Здесь ***countLine*** – глобальная переменная, считающая строки, создающие поля в таблице базы данных. Функция ***addLine()*** реализует добавление полей в таблицу. Эти поля имеют такие свойства, как имя, тип, возможность быть *null*-значением, возможность определять значение по умолчанию и прочее, например, *AUTO\_INCREMENT*.

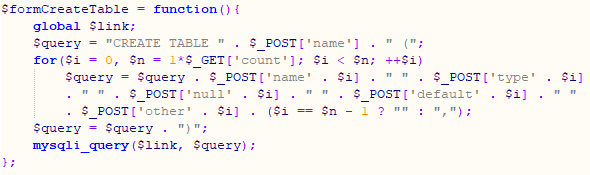
Событие ***window.onload*** позволяет дождаться полной загрузки страницы, после чего кнопке с идентификатором ***addLine*** в обработчик события нажатия присваивается ранее описанная функция ***addLine()***.

**Таким образом**, была реализована необходимая динамичность. Теперь требуется начать написание бэк-энд части данного приложения.

**2.3 Ключевые *PHP*-функции**

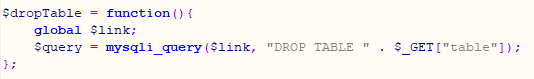
Ниже будет представлена вся подноготная проекта.

Листинг 2.2



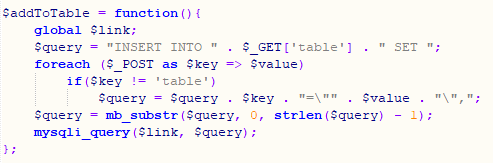
Функция ***$formCreateTable()*** позволяет создавать таблицы по данным из *POST*- запроса.

Листинг 2.3



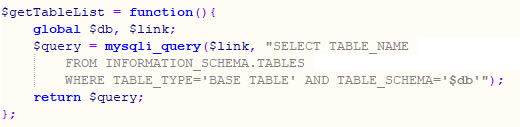
***$dropTable()*** удаляет таблицу по имени из *GET*-запроса.

Листинг 2.4



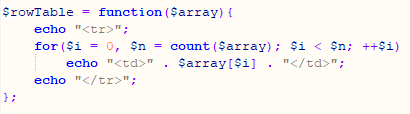
***$addToTable()*** добавляет запись к созданной таблице. Код данной функции позволяет генерировать *SQL*-запрос, основываясь на данных с *POST*-запросов.

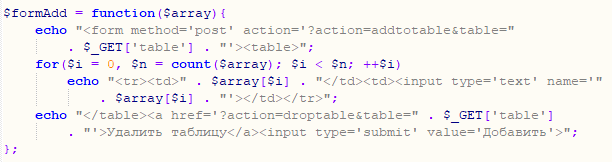
Листинг 2.5

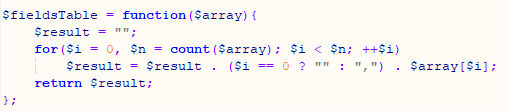


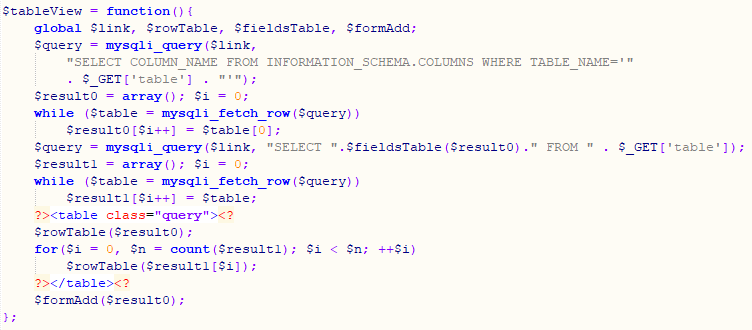
***$getTableList()*** позволяет получить список таблиц из базы данных (*$db*), отведенной для проекта.

Листинг 2.6









И один из самых сложных участков кода.

***$rowTable()*** – выводит строку таблицы с заданными столбцами в аргументе.

***$formAdd()*** – генерирует форму для добавления записи на основе данных о таблице.

***$fieldsTable()*** – объединяет имена полей, обособляя запятыми, в одну строку для *SQL*-запроса, дабы получить на выходе ответ от СУБД, соответствующий порядку полученных имен таблицы.

***$tableView()*** – выводит данные таблицы с формой для создания новых записей для этой таблицы.

**Вывод**: при описании показанных выше листингов внимание уделялось назначению функций и их взаимодействию. Нередко эти функции содержали *SQL*-запросы, ранее не использованные в других лабораторных работах. Теперь стоит приступить к описанию этих запросов.

**2.4 Новые SQL-запросы**

В данной программе использовались SQL-запросы, ранее не требуемые в лабораторных работах. Первый запрос:

Листинг 2.7

CREATE DATABASE $db

Создает базу данных по аргументу ***$db***.

Листинг 2.8

DROP TABLE $db

Удаляет таблицу по аргументу ***$db***.

Листинг 2.9

SELECT TABLE\_NAME

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_TYPE='BASE TABLE' AND TABLE\_SCHEMA='$db'

Выбирает только имена таблиц, которые относятся к базе данных ***$db***.

Листинг 2.10

SELECT COLUMN\_NAME

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME=’$table’

Позволяет получить имена колонок для заданной таблицы ***$table***.

**Вывод**, были приведены SQL-запросы, необходимые для выполнения данной лабораторной работы.

**По итогу главы**, веб-приложение написано.

# ЗАКЛЮЧЕНИЕ

# Был выполнено поставленное задание по изучению веб-интерфейса phpMyAdmin, описанию структуры приложения, а также его реализация на стеке технологий *HTML + CSS + JavaScript + PHP + MySQL*, а значит, что цель отчета выполнена.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. К. Дж. Дейт. Введение в системы баз данных, 2005 г. — 1328 с.
2. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник для высших учебных заведений, 2009 г. – 736 с.
3. <https://metanit.com/sql/mysql/>
4. <https://proselyte.net/tutorials/sql/>