

А. И. Новиков

**АЛГОРИТМИЗАЦИЯ
И ПРОГРАММИРОВАНИЕ
SQL, Apache, PHP, HTML**

Часть 2

Учебно-методическое пособие

**Санкт-Петербург
2025**

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«Санкт-Петербургский государственный университет
промышленных технологий и дизайна»
Высшая школа технологии и энергетики**

А. И. Новиков

**АЛГОРИТМИЗАЦИЯ
И ПРОГРАММИРОВАНИЕ
SQL, Apache, PHP, HTML**

Часть 2

Учебно-методическое пособие

Утверждено Редакционно-издательским советом ВШТЭ СПбГУПТД

Санкт-Петербург
2025

УДК 004.432
ББК 32.973
Н 731

Рецензенты:

кандидат технических наук, доцент, заведующий кафедрой автоматизации технологических процессов и производств Высшей школы технологии и энергетики Санкт-Петербургского государственного университета промышленных технологий и дизайна
Д. А. Ковалёв;

кандидат технических наук, доцент Санкт-Петербургского государственного технологического института (технического университета)
В. В. Куркина

Новиков А. И.

Н 731 Алгоритмизация и программирование. SQL, Apache, PHP, HTML. Часть 2: учебно-методическое пособие / А. И. Новиков. — СПб.: ВШТЭ СПбГУПТД, 2025. — 113 с.

Учебно-методическое пособие соответствует программам и учебным планам дисциплины «Алгоритмизация и программирование» для студентов, обучающихся по направлению подготовки 09.03.03 «Прикладная информатика». В учебно-методическом пособии изложены основы работы с различными СУБД и синтаксис языка SQL. Приведены примеры выполнения SQL-запросов, в том числе с использованием языков PHP, HTML и Delphi (Pascal). Учебно-методическое пособие содержит разделы для самостоятельного углубленного изучения.

Учебно-методическое пособие предназначено для подготовки бакалавров очной и заочной форм обучения. Отдельные разделы могут быть полезны магистрантам и аспирантам.

УДК 004.432
ББК 32.973

© ВШТЭ СПбГУПТД, 2025
© Новиков А. И., 2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	4
5. СЛОЖНЫЕ SQL-ЗАПРОСЫ	5
5.1. Создание страницы	5
5.2. Агрегатные функции и группировка	8
5.3. Объединение таблиц (JOIN)	9
5.4. Пример объединения таблиц (JOIN)	10
5.5. Использование функций IFNULL и COALESCE	13
5.6. Постраничный вывод и ограничение выборки (LIMIT)	14
5.7. РАБОТА № 4 (часть 1 из 2)	15
5.8. Элементы управления в HTML	16
5.9. HTML-формы	17
5.10. Получение значений форм	18
5.11. Динамическое конструирование SQL-запросов	18
5.12. РАБОТА № 4 (часть 2 из 2)	19
5.13. *РАБОТА № 4 (часть 3 из 2)	20
5.14. **РАБОТА № 5	20
5.15. **Переменные SQL	21
6. *SQL В DELPHI	25
6.1. *ADO	26
6.2. *Подключение к MS Access	27
6.3. **ODBC	32
6.4. *Подключение к Excel	33
6.5. *Отображение графиков	39
6.6. **Добавление нового источника данных	41
6.7. **Администрирование ODBC в Windows	45
6.8. **Подключение к MySQL	47
7. *САМОСТОЯТЕЛЬНОЕ УГЛУБЛЕННОЕ ИЗУЧЕНИЕ	53
7.1. *Настройка прав доступа к MySQL	53
7.2. *Оптимизация PHP-кода для MySQL	57
7.3. *Использование MariaDB вместо MySQL	57
7.4. *Файл настроек htaccess	60
7.5. *Отображение содержимого папки (листинг и индекс)	62
7.6. *Настройка кэширования файлов	64
7.7. *Простая переадресация	65
7.8. *Отображение страниц ошибок	66
7.9. **Коды ошибок HTTP (404 и др.)	67
7.10. **Просмотр кодов ответов и HTTP-заголовков в браузере	74
7.11. *Настройка прав доступа к Apache-серверу	76
7.12. *Доступ к серверу с других устройств	78
7.13. **Шифрование и SSL-сертификат	86
7.14. **Доступ к серверу из глобальной сети	87
7.15. *Автозапуск WAMP-сервера	88
7.16. **Arduino и MySQL	92
7.17. **Установка и настройка WAMP-сервера для компьютерного класса ..	98
КОНТРОЛЬНЫЕ ВОПРОСЫ	107
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	109

ВВЕДЕНИЕ

Данное пособие состоит из двух частей. Во второй части изложен материал для углубленного изучения СУБД и языка SQL.

Для объединения таблиц используется оператор JOIN. Он имеет несколько вариантов записи, основные из которых

- INNER JOIN – выбирает записи, имеющие совпадающие значения в обеих таблицах. Записи, не имеющие совпадений, отбрасываются.
- LEFT JOIN – выбирает все записи из левой таблицы и соответствующие ей записи из правой таблицы. Записи из правой таблицы, не имеющие совпадений, отбрасываются.
- RIGHT JOIN – выбирает все записи из правой таблицы и соответствующие ей записи из левой таблицы. Записи из левой таблицы, не имеющие совпадений, отбрасываются.

ODBC – это программный интерфейс (API) доступа к базам данных, разработанный компанией Microsoft. Стандарт призван унифицировать программное взаимодействие с СУБД, сделать его независимым от поставщика СУБД и программно-аппаратной платформы. Драйвер ODBC можно рассматривать как аналог драйвера принтера или другого устройства, предоставляющий стандартный набор функций для использования приложением и реализующий функциональность, специфичную для СУБД. Любое ODBC-совместимое приложение может получить доступ к любой СУБД, для которой установлен драйвер. Драйверы существуют для всех основных СУБД и многих других источников данных, таких как Microsoft Excel или даже для текстовых файлов.

.htaccess (от англ. «hypertext access») – файл дополнительной конфигурации веб-сервера Apache, позволяющий, например, закрыть директорию паролем, запретить доступ к файлам или папкам, исправить кодировку, назначить индексные страницы (страницы по умолчанию), запретить/разрешить выдачу листинга (списка файлов в каталогах), настроить редирект (переадресацию страниц), установить страницы для отображения ошибок (403, 404 и т.п.).

* Пособие содержит разделы для самостоятельного углубленного изучения, их названия отмечены звездочкой.

5. СЛОЖНЫЕ SQL-ЗАПРОСЫ

5.1. Создание страницы

Создадим (пока без использования PHP и MySQL) **HTML**-страницу «**Lab4.html**», в которой добавим список из устройств (пока достаточно 3 шт.), аналогичный представленному на рисунке 141. По каждому из устройств должны отображаться:

- наименование устройства;
- изображение устройства (изображение должно иметь маленький размер, причем для всех устройств необходимо отображать изображения одинаковых размеров);
- цена устройства;
- наличие устройства;
- рейтинг устройства;
- 1-2 основные характеристики устройства.

	ДТС015-PT1000.B2.200 Производитель: Овен Погрешность: 0,5%	Рейтинг: 4,7 Цена: 2 490,00 руб. Наличие: 523 шт.
	ДТС035M-50M.1.0.120.MG.I [3] Производитель: Овен Погрешность: 1%	Рейтинг: 4,9 Цена: 7 998 руб. Наличие: 12 шт.
	ДТС035M-50M.1.0.120.I [3] Производитель: Овен Погрешность: 1%	Рейтинг: 4,9 Цена: 7 398 руб. Наличие: 0 шт.

Рисунок 141 – Страница со списком устройств

Внешний вид может отличаться, и выглядеть, например, как на рисунке 142. Если требуется оформить список как на рисунке 141, то необходимо ознакомиться с разделом 3.3.



[ДТС015-PT1000.B2.200](#)

Производитель: Овен

Погрешность: 0,5%

Рейтинг: 4,7

Цена: 2 490,00 руб.

Наличие: 523 шт.



[ДТС035M-50M.1,0.120.МГ.И \[3\]](#)

Производитель: Овен

Погрешность: 1%

Рейтинг: 4,9

Цена: 7 998 руб.

Наличие: 12 шт.



[ДТС035M-50M.1,0.120.И \[3\]](#)

Производитель: Овен

Погрешность: 1%

Рейтинг: 4,9

Цена: 7 398 руб.

Наличие: 0 шт.

Рисунок 142 – Вариант отображения списка устройств

Также необходимо создать **ссылки** для перехода на страницы соответствующих устройств. Переход происходит при нажатии на **название** устройства или его **изображение**, например, как представлено на рисунке 143.

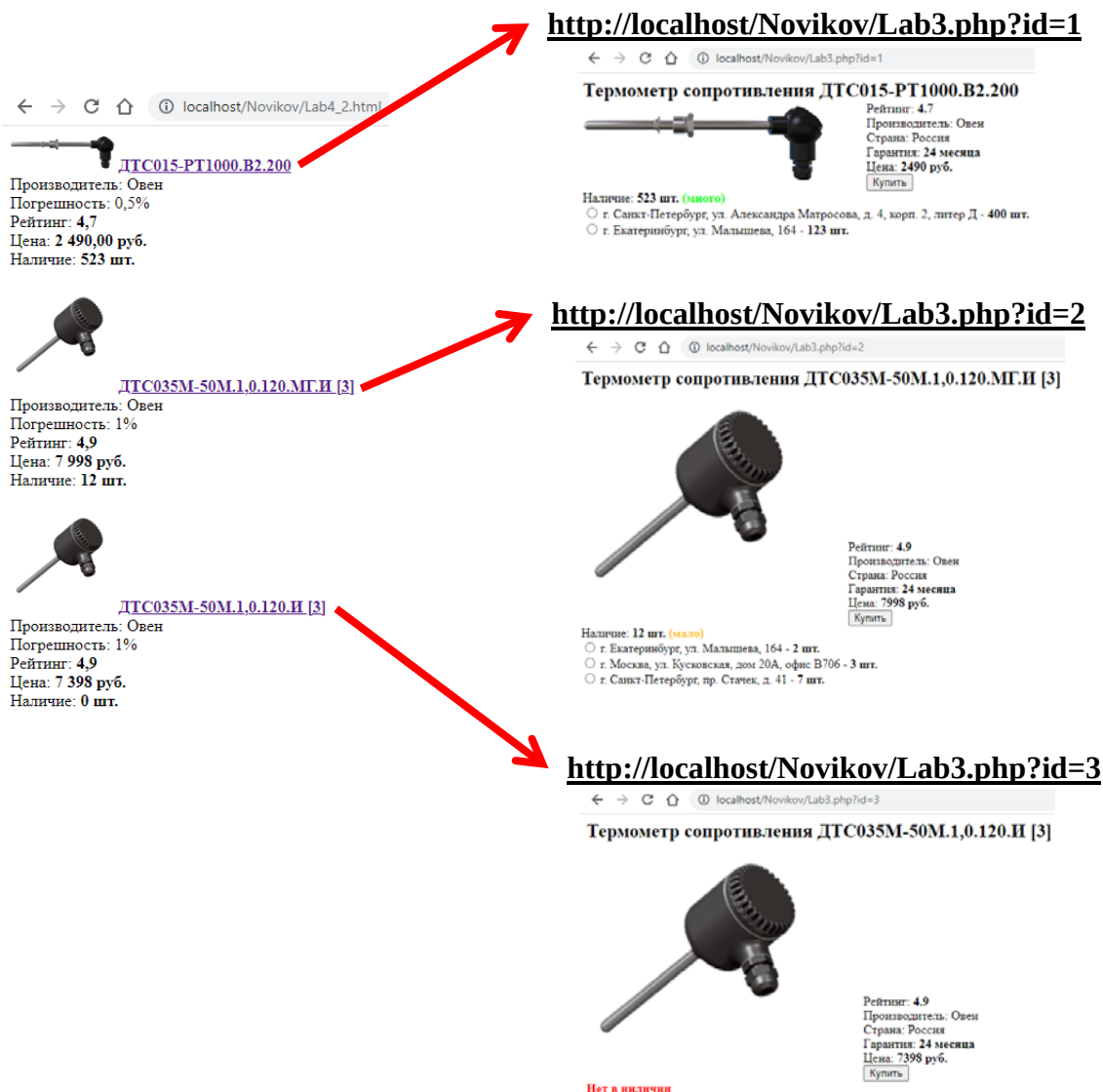


Рисунок 143 – Переходы на страницы устройств

Когда данная страница создана, переименуем файл в «**Lab4.php**» и добавим в его начало код подключения к базе данных **MySQL** (см. раздел 4.5). После этого необходимо реализовать запрос, выводящий устройства в заданном диапазоне цен, выстроенные по рейтингу. Для вывода на страницу нескольких строк результата, оформленных в одном стиле, необходимо использовать циклы, описанные в разделе 4.7.

Стоит обратить внимание, что в отличие от Работы № 3, здесь на страницу выводятся не все данные, имеющиеся в таблице, следовательно, использовать запрос со звездочкой «**SELECT ***» будет неоправданным расточительством. *Хотя это и не будет заметно при выполнении одного запроса раз в несколько секунд, но при выполнении сотен и тысяч запросов в секунду от многих пользователей это значительно снизит производительность сервера.* Требуется вместо звездочки использовать список столбцов, которые участвуют в отображении информации на странице.

Рекомендуется вначале выполнить данную работу без вывода из БД количества шт. в наличии. А потом доработать запрос при помощи операторов **SUM** и **GROUP BY**.

5.2. Агрегатные функции и группировка

Рассмотрим пример таблицы **test3_tbl**, представленный на рисунке 144.

Номер	Имя	Количество
1	Wi-Fi роутер Mercusys MW301R	25
2	Wi-Fi роутер HUAWEI WS5200-21	7
3	Wi-Fi роутер HUAWEI WS5200-21	10
4	Wi-Fi роутер Mercusys MW301R	5
5	Wi-Fi роутер TP-LINK TL-WR844N	27

Рисунок 144 – Исходная таблица

Для определения суммарного значения применяется агрегатная функция **SUM**, которая используется в запросе следующим образом:

```
SELECT SUM(Количество)
FROM test3_tbl;
```

Результат выполнения данного запроса представлен на рисунке 145.

SUM(Количество)
74

Рисунок 145 – Результат суммирования

При таком способе невозможно вывести содержимое других столбцов (без агрегатных функций), например, запрос:

```
SELECT Номер, Имя, SUM(Количество)
FROM test3_tbl;
```

вернет результат, представленный на рисунке 146, который не соответствует действительности, т.к. **Количество** посчитано по всем устройствам, а **Имя** и **Номер** отображены только для первого из них, что конечно же неверно.

Номер	Имя	SUM(Количество)
1	Wi-Fi роутер Mercusys MW301R	74

Рисунок 146 – Неверный подсчет количества

Обычно **SUM** используется совместно с **GROUP BY**:

```
SELECT Имя, SUM(Количество)
FROM test3_tbl
```

GROUP BY Имя;

Здесь мы можем вывести и столбцы без агрегатных функций, но только те столбцы, по которым ведется группировка! Результат такого запроса представлен на рисунке 147.

Имя	SUM(Количество)
Wi-Fi роутер HUAWEI WS5200-21	17
Wi-Fi роутер Mercusys MW301R	30
Wi-Fi роутер TP-LINK TL-WR844N	27

Рисунок 147 – Суммирование по группам

Агрегатную функцию **SUM** невозможно использовать в условиях **WHERE**, например, нельзя написать:

```
WHERE SUM(Количество) > 20
```

Аналогом **WHERE** для агрегатных функций является **HAVING**:

```
SELECT Имя, SUM(Количество)
```

```
FROM test3_tbl
```

```
GROUP BY Имя
```

```
HAVING SUM(Количество) > 20;
```

В этом случае мы получим результат, представленный на рисунке 148.

Имя	SUM(Количество)
Wi-Fi роутер Mercusys MW301R	30
Wi-Fi роутер TP-LINK TL-WR844N	27

Рисунок 148 – Условие HAVING

Аналогично **SUM** можно использовать и другие агрегатные функции **MIN**, **MAX**, **AVG** и **COUNT** – для поиска минимального, максимального и среднего значения, а также для определения количества найденных записей (строк).

5.3. Объединение таблиц (JOIN)

Оператор **JOIN** используется для объединения таблиц. Виды объединения таблиц представлены на рисунке 149.

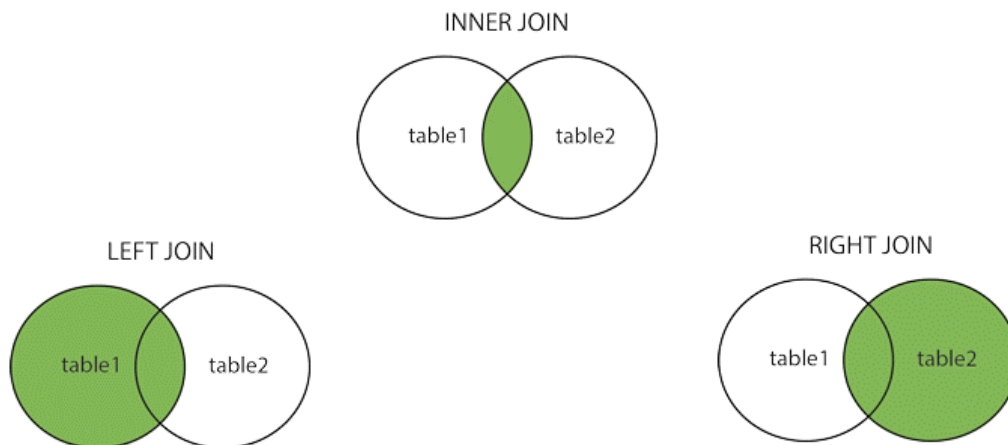


Рисунок 149 – Виды объединения таблиц

INNER JOIN выбирает записи, имеющие совпадающие значения в обеих таблицах[1]. *Записи, не имеющие совпадений, отбрасываются.*

FROM Таблица1 **INNER JOIN** Таблица2

ON Таблица1.Имя_столбца = Таблица2.Имя_столбца

Оператор **JOIN** указывается после оператора **FROM**, фактически, вместо запятой в перечислении таблиц. Ранее мы писали:

FROM Таблица1, Таблица2

Оператор **JOIN** используется совместно с оператором **ON**, после которого указываются имена связываемых столбцов (аналогично тому, как это делалось при связывании через **WHERE**).

LEFT JOIN (он же **LEFT OUTER JOIN**) выбирает **все** записи из левой таблицы (Таблица1) и соответствующие ей записи из правой таблицы (Таблица2) [2]. *Записи из правой таблицы (Таблица2), не имеющие совпадений, отбрасываются.*

FROM Таблица1 **LEFT JOIN** Таблица2

ON Таблица1.Имя_столбца = Таблица2.Имя_столбца

RIGHT JOIN (он же **RIGHT OUTER JOIN**) выбирает **все** записи из правой таблицы (Таблица2) и соответствующие ей записи из левой таблицы (Таблица1) [3]. *Записи из левой таблицы (Таблица1), не имеющие совпадений, отбрасываются.*

FROM Таблица1 **RIGHT JOIN** Таблица2

ON Таблица1.Имя_столбца = Таблица2.Имя_столбца

5.4. Пример объединения таблиц (JOIN)

Рассмотрим пример двух таблиц, **test4_1** (рис. 150) и **test4_2** (рис. 151).

Номер	Наименование	...
1	Wi-Fi роутер Mercusys MW301R	...
2	Wi-Fi роутер HUAWEI WS5200-21	...
3	Wi-Fi роутер TP-LINK TL-WR844N	...
4	Коммутатор D-Link DES-1005C/B	...
5	Коммутатор Zyxel GS-108S v2	...

Рисунок 150 – Пример таблицы с устройствами

No	Номер устройства	Город	Количество	Цена	...
1	1	Санкт-Петербург	37	1199	...
2	1	Москва	54	1099	...
3	1	Мурманск	3	1308	...
4	2	Санкт-Петербург	0	2799	...
5	3	Москва	7	1599	...

Рисунок 151 – Пример таблицы с количеством устройств

Если по этим таблицам выполнить следующий запрос с **INNER JOIN**, то получим таблицу, представленную на рисунке 152 (зеленой вертикальной линией показана граница между двумя исходными таблицами):

```
SELECT *
FROM test4_1 INNER JOIN test4_2
ON test4_1.Номер = test4_2.`Номер устройства`;
```

Номер	Наименование	...	No	Номер устройства	Город	Количество	Цена	...
1	Wi-Fi роутер Mercusys MW301R	...	1	1	Санкт-Петербург	37	1199	...
1	Wi-Fi роутер Mercusys MW301R	...	2	1	Москва	54	1099	...
1	Wi-Fi роутер Mercusys MW301R	...	3	1	Мурманск	3	1308	...
2	Wi-Fi роутер HUAWEI WS5200-21	...	4	2	Санкт-Петербург	0	2799	...
3	Wi-Fi роутер TP-LINK TL-WR844N	...	5	3	Москва	7	1599	...

Рисунок 152 – Пример запроса INNER JOIN

Данная таблица получена путем объединения двух исходных таблиц по схеме, представленной на рисунке 153.

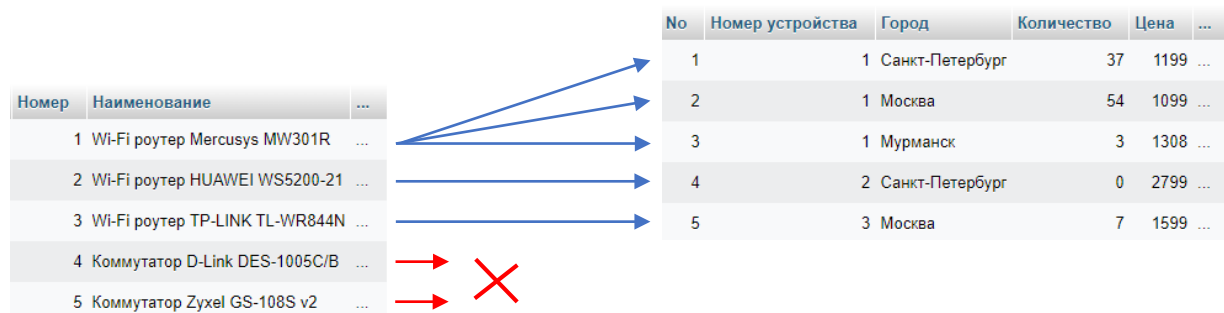


Рисунок 153 – Схема объединения таблиц через INNER JOIN

Из первой таблицы (**test4_1**) берутся строки с номерами 1, 2 и 3 (некоторые даже не по одному разу), т.к. ссылки на них присутствуют во второй таблице (**test4_2**). Устройства с номерами 4 и 5 отбрасываются, т.к. для них нет соответствующих записей во второй таблице (**test4_2**).

Если выполнить тот же запрос, но с **LEFT JOIN**, то получим таблицу, представленную на рисунке 154:

```
SELECT *
FROM test4_1 LEFT JOIN test4_2
ON test4_1.Номер = test4_2.`Номер устройства`;
```

Номер	Наименование	...	No	Номер устройства	Город	Количество	Цена	...
1	Wi-Fi роутер Mercusys MW301R	...	1	1	Санкт-Петербург	37	1199	...
1	Wi-Fi роутер Mercusys MW301R	...	2	1	Москва	54	1099	...
1	Wi-Fi роутер Mercusys MW301R	...	3	1	Мурманск	3	1308	...
2	Wi-Fi роутер HUAWEI WS5200-21	...	4	2	Санкт-Петербург	0	2799	...
3	Wi-Fi роутер TP-LINK TL-WR844N	...	5	3	Москва	7	1599	...
4	Коммутатор D-Link DES-1005C/B	...	NULL	NULL	NULL	NULL	NULL	NULL
5	Коммутатор Zyxel GS-108S v2	...	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 154 – Пример запроса LEFT JOIN

Здесь мы видим, что из первой таблицы были подставлены **все** возможные строки (некоторые даже не по одному разу). Из второй таблицы были подставлены строки, имеющие совпадения с первой таблицей. Недостающие значения в правой части (показаны красной рамкой) были заполнены значениями **NULL**.

Аналогично, используя **агрегатные функции** и **группировку**, можно выполнить следующий запрос с **INNER JOIN** (рис. 155):

```
SELECT test4_1.Номер, Наименование,
       sum(Количество) AS `Кол-во`, min(Цена) AS `Мин. цена`
FROM test4_1 INNER JOIN test4_2
ON test4_1.Номер = test4_2.`Номер устройства`
GROUP BY Номер
ORDER BY sum(Количество) DESC;
```

Номер	Наименование	Кол-во	Мин. цена
1	Wi-Fi роутер Mercusys MW301R	94	1099
3	Wi-Fi роутер TP-LINK TL-WR844N	7	1599
2	Wi-Fi роутер HUAWEI WS5200-21	0	2799

Рисунок 155 – Пример запроса INNER JOIN с группировкой

Проблемой является то, что устройства с номерами 2, 4 и 5 отсутствуют в наличии в нашем магазине (их 0 шт.), но при этом устройство с номером 2 попало в таблицу с результатами (т.к. в исходных таблицах для него явно указано количество **0**), а устройства с номерами 4 и 5 не попали (т.к. в исходных таблицах для них нет записей с количеством).

Тот же запрос, использующий **агрегатные функции** и **группировку**, но с **LEFT JOIN**, будет выглядеть следующим образом (рис. 156):

```
SELECT test4_1.Номер, Наименование,
       sum(Количество) AS `Кол-во`, min(Цена) AS `Мин. цена`
FROM test4_1 LEFT JOIN test4_2
ON test4_1.Номер = test4_2.`Номер устройства`
GROUP BY Номер
ORDER BY sum(Количество) DESC;
```

Номер	Наименование	Кол-во	Мин. цена
1	Wi-Fi роутер Mercusys MW301R	94	1099
3	Wi-Fi роутер TP-LINK TL-WR844N	7	1599
2	Wi-Fi роутер HUAWEI WS5200-21	0	2799
4	Коммутатор D-Link DES-1005C/B	NULL	NULL
5	Коммутатор Zyxel GS-108S v2	NULL	NULL

Рисунок 156 – Пример запроса LEFT JOIN с группировкой

Были выведены все устройства, но и здесь имеется проблема: часть значений для отсутствующих в продаже устройств отображается как **0**, а часть – как **NULL**.

5.5. Использование функций IFNULL и COALESCE

Функция **IFNULL** принимает на вход два параметра, возвращая первый из них, только если он не нулевой ($\neq \text{NULL}$). Если первый аргумент **NULL**, то возвращается второй аргумент.

Таким образом, мы можем использовать данную функцию для присвоения значения **по умолчанию** (например, значения **0**, **'нет'** или пустой строки **''**) в случае, если база данных выдает значение **NULL**. *Причем данные*

манипуляции осуществляются на стороне **SQL**-сервера, и нет необходимости в дополнительных проверках (**IF**) полученных значений на стороне **PHP**, или на стороне клиента (**JavaScript**).

Будем использовать эту функцию в шапке запроса **SELECT**, где идет перечисление запрашиваемых столбцов, например, вместо:

```
SELECT ... sum(Количество) ... min(Цена) ...
```

теперь используем:

```
SELECT ... IFNULL(sum(Количество), 0) ... IFNULL(min(Цена), "") ...
```

Тогда в итоге доработанный предыдущий запрос вернет нам уже не значения **NULL**, а подставит значения **0** (рис. 157) или пустую строку "".

Номер	Наименование	Кол-во	Мин. цена
1	Wi-Fi роутер Mercusys MW301R	94	1099
3	Wi-Fi роутер TP-LINK TL-WR844N	7	1599
2	Wi-Fi роутер HUAWEI WS5200-21	0	2799
4	Коммутатор D-Link DES-1005C/B	0	
5	Коммутатор Zyxel GS-108S v2	0	

Рисунок 157 – Итоговая таблица

Функция **COALESCE** принимает на вход два или более параметра, возвращая первый ненулевой ($\neq \text{NULL}$) из них [4]. При двух аргументах **COALESCE** аналогична функции **IFNULL**.

!!! Функция **COALESCE** является стандартной функцией **SQL**, в то время как **IFNULL** является функцией только **MySQL**. Таким образом, если необходимо писать код, который должен поддерживать несколько различных версий **SQL** (несколько различных СУБД), необходимо использовать именно **COALESCE**. Т.е. вместо предыдущего варианта лучше написать:

```
SELECT ... COALESCE(sum(Количество), 0) ...  
... COALESCE(min(Цена), "") ...
```

!!! Функции **IFNULL** и **COALESCE** могут применяться не только совместно с объединением таблиц (**JOIN**) или агрегатными функциями (**SUM**, **COUNT**, **AVG**, **MIN**, **MAX**). Столбцы, которые не помечены как **NOT NULL**, также могут содержать значения **NULL**. Таким образом, для применения данной функции достаточно и всего одной таблицы.

5.6. Постраничный вывод и ограничение выборки (**LIMIT**)

Запрос может выдавать сотни и тысячи строк результатов. При этом все результаты могут не поместиться на одной странице, либо вывод всех результатов может быть нецелесообразен (например, товары отсортированы по возрастанию цены, тогда пользователя могут просто не интересовать самые дорогие из товаров, которые будут расположены в конце таблицы с результатами).

Для ограничения выборки используется ключевое слово **LIMIT**. Например, следующий запрос выдаст только **первые 3 строки** с результатами:

```
SELECT *  
FROM test3_tbl  
LIMIT 3;
```

Ключевое слово **LIMIT** может использоваться и с двумя параметрами:

LIMIT количество_пропущенных_записей, вывести_записей
или **LIMIT** вывести_записей **OFFSET** количество_пропущенных_записей

Например, следующие записи выводят строки результатов с 21 по 30:

LIMIT 20, 10
или **LIMIT** 10 **OFFSET** 20

Причем запись с одним параметром, имеющая вид:

LIMIT 10

равносильна записям:

LIMIT 0, 10
или **LIMIT** 10 **OFFSET** 0

Стоит отметить, что ключевое слово **LIMIT**, является командой не **SQL**, а только **MySQL** (а также **MariaDB**). В *СУБД других производителей, тот же функционал реализуется по-другому, например, командой **SELECT TOP**, или при помощи системной переменной **ROWNUM**, записываемой в условии **WHERE**.*

5.7. РАБОТА № 4 (часть 1 из 2)

«Сложные SQL-запросы»

Требуется создать **PHP**-страницу, аналогичную описанной в разделе 5.1. При этом страница должна содержать ссылки на страницы, разработанные в Работе № 3 (с разными **id**).

Данные для отображения страницы должны запрашиваться из БД (созданной в Работе № 2). Необходимо получить все требуемые данные одним запросом.

!!! В этой работе запрещается использовать символ звездочки «*» в запросах **Select**!

В первой части данной работы достаточно использовать жестко прописанные в коде запросы. При этом необходимо проверить работоспособность как минимум для двух случаев:

- вывести устройства в заданном диапазоне цен, выстроенные по рейтингу;
- вывести устройства с заданной погрешностью (либо указанного производителя, если погрешность для данных устройств

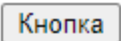
неприменима), имеющиеся в наличии, выстроенные по возрастанию цены.

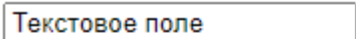
!!! В базе данных должно иметься хотя бы два устройства, которых нет в наличии (причем для одного должно быть явно указано 0 шт., а для другого должны отсутствовать записи о количестве). Для этого может понадобиться отредактировать данные в таблицах, созданных в Работе № 2.

5.8. Элементы управления в HTML

Кроме элемента `<button>` – кнопка (рис. 158а), язык **HTML** содержит и другие элементы управления, перечисленные ниже.

1. Поле для ввода текста – рисунок 158б: `<input type="text">`.
2. Флажок (галочка, чекбокс) – рисунок 158в: `<input type="checkbox">`.
3. Радиокнопка (переключатель) – рисунок 158г: `<input type="radio">`.
4. Текстовая область с многострочным текстом – рисунок 158д: `<textarea>`.
5. Однострочный список – рисунок 158е: `<select size=1>`.
6. Многострочный список – рисунок 158ё: `<select size=3>`.
7. Многострочный список с множественным выбором – рисунок 158ж: `<select size=3 multiple>`.

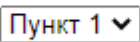
а)  Кнопка

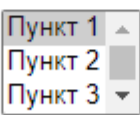
б)  Текстовое поле

в)  Флажок

г)  Радиокнопка 1  Радиокнопка 2  Радиокнопка 3

д)  Текстовая область с многострочным текстом

е) Однострочный список  Пункт 1 ▼

ё) Многострочный список  Пункт 1 ▲
Пункт 2
Пункт 3 ▼

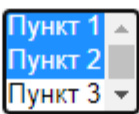
ж) С множественным выбором  Пункт 1 ▲
Пункт 2
Пункт 3 ▼

Рисунок 158 – Элементы управления

Код, по которому были созданы элементы управления на рисунке 158, представлен далее:

```
а) <button>Кнопка</button> <br><br>
б) <input type="text" value="Текстовое поле"> <br><br>
в) <input type="checkbox" checked>Флажок <br><br>
г) <input type="radio" name="aaa" value="val1" checked>Радиокнопка
1
    <input type="radio" name="aaa" value="val2">Радиокнопка 2
    <input type="radio" name="aaa" value="val3">Радиокнопка 3
    <br><br>
д) <textarea cols=70 rows=5>Текстовая область с
многострочным текстом</textarea> <br><br>
е) Однострочный список
<select size=1>
    <option selected>Пункт 1</option>
    <option>Пункт 2</option>
</select> <br><br>
ё) Многострочный список
<select size=3>
    <option selected>Пункт 1</option>
    <option>Пункт 2</option>
    <option>Пункт 3</option>
    <option>Пункт 4</option>
</select> <br><br>
ж) С множественным выбором
<select size=3 multiple>
    <option selected>Пункт 1</option>
    <option selected>Пункт 2</option>
    <option>Пункт 3</option>
    <option>Пункт 4</option>
</select>
```

5.9. HTML-формы

Для того, чтобы элементы управления выполняли свою функцию, необходимо «обернуть» их тегами **<form>** и **</form>**, например:

```
<form>
    <select size=1 name="sort">
        <option value="rating">По рейтингу</option>
        <option value="price" selected>По возрастанию цены</option>
        ...
    </select>

    <input type="checkbox" name="available" checked>В наличии
    ...
    <input type="text" name="name">
    <button>Поиск</button>
</form> <br>
```

В таком случае, после нажатия кнопки **button** мы увидим изменения в адресной строке, в которой отобразятся введенные параметры запроса (рис. 159).

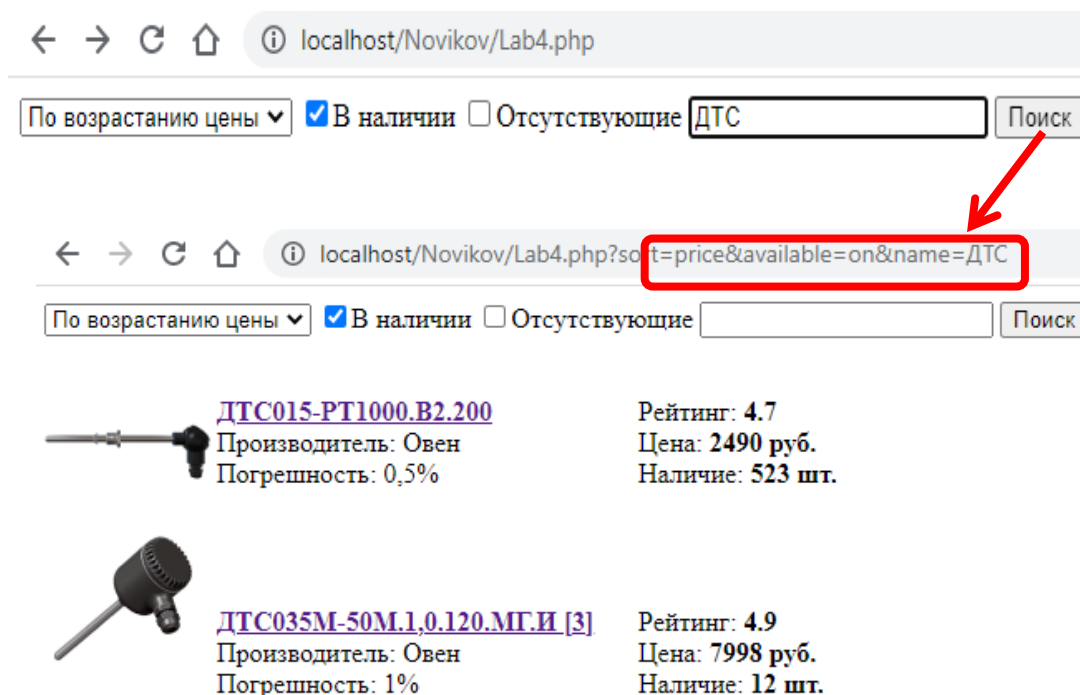


Рисунок 159 – Передача параметров через адресную строку

Стоит обратить внимание, что в этом случае атрибут **name** становится **обязательным** для всех элементов управления, т.к. именно эти имена отображаются в адресной строке.

5.10. Получение значений форм

Получение значений из адресной строки уже упоминалось в разделе 4.9. Рассмотрим его подробнее:

```
$id = !empty($_GET['id']) ? $_GET['id'] : 1;
```

Здесь **\$id** – имя переменной, в которую сохраняется значение;

'id' – имя параметра в адресной строке;

1 – значение по умолчанию, присваиваемое переменной **\$id** в случае, когда значение для **'id'** в адресной строке не задано.

Для каждого из параметров, передаваемых через адресную строку, необходимо создать свою переменную.

5.11. Динамическое конструирование SQL-запросов

Трудно представить себе, чтобы сложный запрос можно было составить заранее (статически). Эти запросы должны конструироваться динамически, путем объединения требуемых строк при выполнении определенных условий. Для объединения строк в **PHP** используется **точка**. Условия записываются при помощи **if**.

Например, для добавления к запросу строки с сортировкой по возрастанию цены, можно использовать следующий **PHP**-код:

```
//Сортировка по возрастанию цены  
if ($sort == 'price') {$sql = $sql . " ORDER BY lab2.`Цена`";}
```

В данном случае переменная **\$sort** со значением «**price**» должна быть получена из адресной строки (см. раздел 5.10).

5.12. РАБОТА № 4 (часть 2 из 2)

«Сложные SQL-запросы»

Требуется дополнить разработанную страницу формой поиска, например, как на рисунке. 160, передающей введенные параметры через адресную строку. Добавить к форме поля, требуемые для выполнения отчета (см. содержание отчета ниже).

По возрастанию цены ▾ ☒ В наличии ☐ Отсутствующие

По рейтингу
По возрастанию цены
По убыванию цены

PT1000.B2.200 Рейтинг: 4.7
Цена: 2490 руб.
Наличие: 523 шт.

Производитель: Овен
Погрешность: 0,5%

Рисунок 160 – Пример формы поиска

Дополнить **PHP**-код вводом параметров поиска через адресную строку (через **\$_GET** – см. разделы 4.9 и 5.10), и реализовать использование этих параметров в SQL-запросе.

Значения в элементах управления, доступные для выбора при поиске, могут быть вбиты «жестко» (т.е. их не нужно загружать из базы данных).

Отчет должен содержать:

- цель и задачи;
- скриншоты таблиц (из Работы № 2, возможно немного доработанных) по которым осуществляются запросы;
- описание хода работы (Что, как и в какой программе делалось? Со скриншотами);
- описание новых тегов HTML (использованных элементов управления и их атрибутов); описание использованных команд SQL и PHP;
- код созданных страниц (HTML+PHP+SQL) с выделением синтаксиса разными цветами и скриншоты созданных страниц, на которых видны **заполненная** форма поиска и результат выполнения этого запроса:
 - устройства в заданном диапазоне цен, выстроенные по рейтингу;
 - устройства с заданной погрешностью (либо указанного производителя, если погрешность для данных устройств неприменима), имеющиеся в наличии, выстроенные по возрастанию цены;

- устройства, которых нет в наличии;
 - один запрос по выбору автора работы;
 - страница с сообщением «По вашему запросу ничего не найдено» (для случая в котором по запросу было найдено **0** строк);
- описание параметров адресной строки;
- титульный лист, номера страниц, оглавление, список использованной литературы (включая Интернет-ресурсы и данную методичку), выводы/заключение и т.п.

5.13. *РАБОТА № 4 (часть 3 из 2)

«Сложные SQL-запросы»

В проделанной ранее Работе, необходимо модернизировать следующее:

- сохранять между запросами значения, заданные в элементах управления. Т.е. не очищать (повторно заполнять) поля ввода после нажатия кнопки «Поиск»;
- изменить разработанную страницу так, чтобы значения в элементах управления были вбиты не «жестко», а подтягивались автоматически из вариантов, доступных в базе данных. Например, если в базе имеются устройства с корпусами, выполненными из пластика и из металла, то в выпадающем списке будет указано «пластик» и «металл». Если далее в базу будет добавлено устройство из дерева, то в выпадающем списке автоматически появится вариант «дерево»;
- дополнить работу страницей сравнения двух устройств. Причем должна иметься возможность **скрыть одинаковые** для обоих устройств значения и отобразить только **отличия**.

В связи с тем, что предметом изучения является **SQL**, а не **PHP**, соответственно, при решении задачи необходимо максимально использовать возможности **SQL**, а не **PHP**.

В настоящем разделе содержатся дополнения к Работе № 4, предназначенные для углубленного изучения, не являющиеся обязательным для выполнения всеми студентами.

5.14. **РАБОТА № 5

«Работа с файлами в PHP»

Разработать несколько служебных модулей **PHP**, расположенных в папке «**Admin**», выполняющих следующие полезные функции:

- после запуска, программа проверяет «битые ссылки» на изображения. Если в таблице (из Работы № 2) содержатся ссылки на изображения, которых не существует в папке «**Img**», то выводится список этих файлов и номера строк в таблице (**id** датчика), которые ссылаются на несуществующие файлы. Если «битых ссылок» нет, то выводится соответствующее сообщение об успешной работе;

- предыдущий пункт лишь помогал определить проблему, но не исправлял ее. Далее нужно разработать программу, которая заменяет все «битые ссылки», на ссылку на одно изображение «**Null.png**». Также необходимо подготовить данное изображение, на котором в стилизованном виде будет написано «Нет изображения» (или т.п.);
- обратной задачей является проверка неиспользуемых изображений в папке «**Img**». Если на какое-то из изображений больше нет ссылки в базе данных (таблице из Работы № 2), то его нужно удалить из папки «**Img**». Стоит также обратить внимание, что на файл «**Null.png**» может не быть действующих ссылок, но его в любом случае **удалять не нужно**.

5.15. **Переменные SQL

В MySQL можно использовать **переменные**, хотя в большинстве случаев это и не рекомендуется (*именно это и является причиной того, что мы рассматриваем переменные только в конце изучения SQL*).

Переменные пользователя

Переменные пользователя должны начинаться с символа «@», они действуют только внутри одного сеанса и при повторных обращениях не сохраняются. Для присваивания значения переменной используется ключевое слово **SET**. При этом мы можем поместить результат запроса в переменную, а дальше использовать его в другом запросе, например:

#Определяем среднее количество сданных работ

```
SET @AvgLab = (SELECT AVG(`Сдано работ`) FROM `Students_Lab`);
```

#Выводим студентов, обучающихся ниже среднего

```
SELECT * FROM `Students_Lab` WHERE `Сдано работ` < @AvgLab;
```

Кроме того, мы в любой момент можем увидеть значение переменной, выполнив команду:

```
SELECT @AvgLab;
```

Сохраняемый запрос при этом заключается в круглые скобки. Такой запрос **SELECT** должен возвращать только **одно** значение (**одну** ячейку таблицы), поэтому:

- в запросе нужно указать **одно** конкретное имя колонки и нельзя использовать вместо него звездочку ***** (*конечно за исключением случая, когда в исходной таблице всего одна колонка*);
- запрос должен возвращать только **одну** строку, поэтому чаще всего используется вместе с агрегатными функциями (**MIN**, **MAX**, **SUM**, **COUNT**, **AVG**).

!!! Этим рассмотренное сохранение в **переменную** практически аналогично (простым) Подзапросам, с той лишь разницей, что промежуточный результат имеет конкретное имя.

Можно сохранить и сразу несколько колонок таблицы в разные переменные (но строка по-прежнему должна быть одна!). Оператор SET здесь не используется:

#Определяем минимум и максимум

```
SELECT @Min := MIN('Сдано работ'), @Max := MAX('Сдано работ')  
FROM `Students_Lab`;
```

!!! В этом случае для присвоения значения применяется не «=», а «:=», т.к. обычный знак равенства уже занят в запросах для оператора сравнения. *При этом с оператором SET, на самом деле, можно использовать любой из вариантов «=» или «:=».*

!!! Данный способ без SET не рекомендуется к применению т.к. считается устаревшим, и его поддержка будет прекращена в следующих версиях MySQL.

Сами по себе **переменные пользователя** являются удобным дополнительным инструментом для написания программ. Но не стоит массово и бездумно использовать сохранение результатов запросов в переменные – для этого лучше использовать сложные SQL-запросы или (простые) Подзапросы. Причина тому кроется в **декларативности** языка SQL (отвечает на вопрос «**Что делать?**»), а указание последовательности сохранения переменных наоборот является **императивным** (отвечает на вопрос «**Как делать?**»). Таким образом, программа SQL написанная с использованием переменных, будет фактически понижена до языка программирования 3-го поколения (**3GL**), хотя SQL является языком 4-го поколения (**4GL**)... Но только если им правильно пользоваться.

!!! В студенческих работах присвоение результатов запросов **переменным пользователя запрещено!**

Параметризация запросов

А вот где стоит использовать **переменные пользователя**, так это для присвоения им **константных** значений (используемых в качестве входных параметров для следующего за ними запроса).

Рассмотрим следующий запрос:

```
SELECT SUM('Сдано работ') AS "Сдано работ", 'Фамилия', 'Имя', 'Отчество'  
FROM Students_Lab  
WHERE ('Предмет' IN ('АиП', 'ОССиТК')) AND ('Сдано работ' <= 5)  
GROUP BY 'Фамилия', 'Имя', 'Отчество'  
HAVING SUM('Сдано работ') BETWEEN 1 AND 5  
ORDER BY SUM('Сдано работ') DESC, 'Фамилия' ASC, 'Имя' ASC, 'Отчество' ASC  
LIMIT 20 OFFSET 0;
```

Этот запрос можно описать следующим образом: «Сформировать список студентов, выполнивших суммарно по двум предметам (АиП и ОССиТК) от 1 до 5 работ, но при этом не выполнивших сразу все 5 работ по любому из этих предметов. Список сортируется по сумме работ, начиная с наибольшего их количества. Студенты с равным количеством работ выстраиваются в алфавитном порядке. Отобразить только первую страницу из 20 человек».

А теперь рассмотрим вариант, при котором нужно выполнить несколько однотипных запросов, но каждый раз с разными «от» и «до». В прошлом

варианте для этого потребовалось бы каждый раз искать требуемые значения в тексте запроса чтобы заменить их. Но гораздо проще вынести эти значения в качестве входных параметров в начало кода.

Тогда, заменив требуемые значения на **переменные**, получим следующее:

```
SET @ot = 1, @do = 5;
SELECT SUM(`Сдано работ`) AS "Сдано работ", `Фамилия`, `Имя`, `Отчество`
FROM Students_Lab
WHERE (`Предмет` IN ('АиП', 'ОССиТК')) AND (`Сдано работ` <> 5)
GROUP BY `Фамилия`, `Имя`, `Отчество`
HAVING SUM(`Сдано работ`) BETWEEN @ot AND @do
ORDER BY SUM(`Сдано работ`) DESC, `Фамилия` ASC, `Имя` ASC, `Отчество` ASC
LIMIT 20 OFFSET 0;
```

Особенно удобно использовать **переменные пользователя** для того, чтобы вынести из запроса длинные текстовые строки, или когда мы хотим изменить порядок ввода значений. Например, исходный запрос:

```
UPDATE Organizations
SET Region='по Санкт-Петербургу и Ленинградской области'
WHERE Region='СПб и ЛО';
```

имеет смысл переписать как:

```
SET @from = 'СПб и ЛО',
    @to = 'по Санкт-Петербургу и Ленинградской области';
UPDATE Organizations SET Region=@to WHERE Region=@from;
```

****Системные переменные**

Системные переменные (Глобальные переменные) начинаются с пары символов «@@», они действуют **не** только внутри одного сеанса и сохраняются при повторных обращениях (в отличие от **переменных пользователя**). Список системных переменных предопределен и не предназначен для пополнения. Для того чтобы вывести полный список системных переменных, необходимо выполнить SQL-запрос:

```
SHOW VARIABLES;
```

Variable_name	Value
connect_timeout	10
delayed_insert_timeout	300
have_statement_timeout	YES
innodb_flush_log_at_timeout	1
innodb_lock_wait_timeout	600
innodb_rollback_on_timeout	OFF
interactive_timeout	28800
lock_wait_timeout	31536000
net_read_timeout	30
net_write_timeout	60
replica_net_timeout	60

Рисунок 161 – Список системных переменных

Для того чтобы вывести список (рис. 161) системных переменных, содержащих только заданную (заранее известную) часть имени, необходимо выполнить запрос вида:

```
SHOW VARIABLES LIKE '%timeout%';
```

Если имя системной переменной известно, то ее значение можно вывести и через запрос SELECT, например:

```
SELECT @@connect_timeout;
```

Для того чтобы изменить значение системной (глобальной) переменной, необходимо выполнить команду вида:

```
SET GLOBAL connect_timeout = 12;
```

или, что тоже-самое:

```
SET @@GLOBAL.connect_timeout = 12;
```

!!! Не рекомендуется изменять значения системных переменных, т.к. это повлияет на работу сервера!

6. *SQL В DELPHI

Delphi (а также **C++ Builder**) позволяет подключаться к различным базам данных с использованием SQL-запросов.

Визуальные компоненты для отображения информации, полученной из базы данных, расположены на вкладке «**Data Controls**» (рис. 162). Далее мы будем рассматривать только два компонента, для отображения таблиц и для отображения графиков. В новых версиях Delphi также есть вкладка «**Data Controls**» (рис. 163), но компонент для работы с графиками расположен не на ней, а на вкладке «**TeeChart Lite**» (рис. 164), вместе с другими графиками.

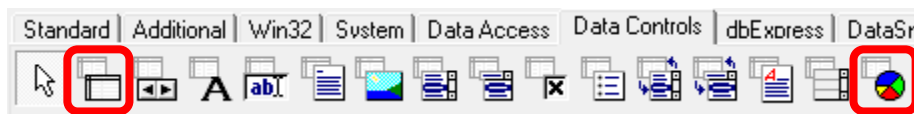


Рисунок 162 – Компоненты для отображения информации в Delphi 7

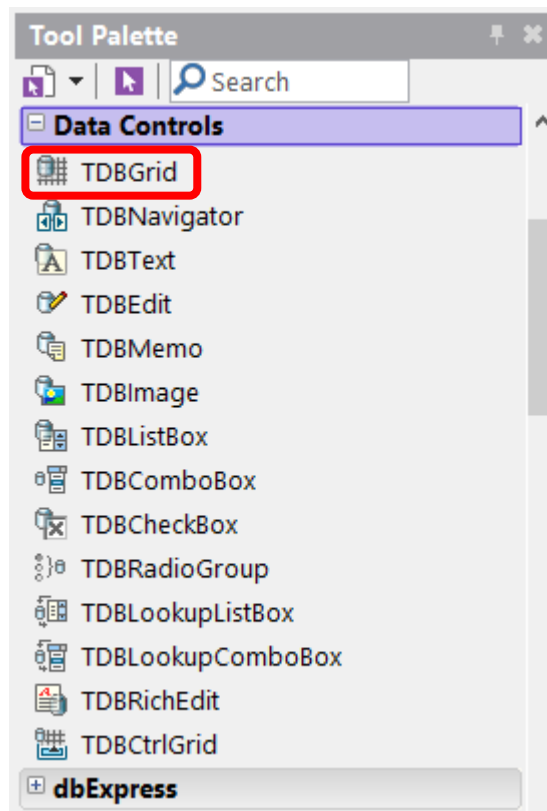


Рисунок 163 – Компонент для отображения таблиц в Delphi XE8

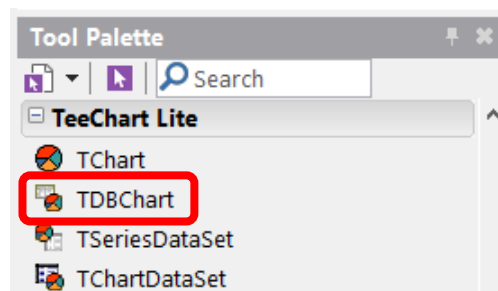


Рисунок 164 – Компонент для отображения графиков в Delphi XE8

Непосредственно подключение к базе данных и выполнение SQL-запросов осуществляется через невидимые компоненты, о которых будет рассказано далее. Сейчас рассмотрим лишь компонент **DataSource**, расположенный на вкладке «**Data Access**» (рис. 165 –166).

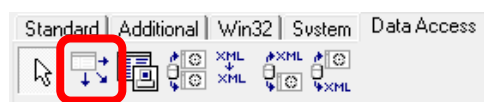


Рисунок 165 – Компонент DataSource в Delphi 7

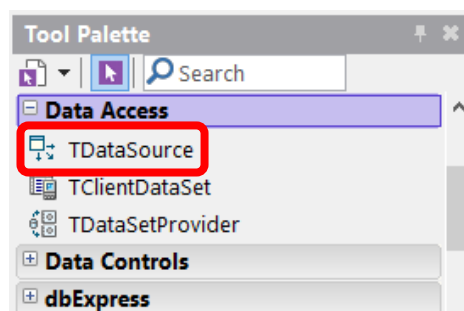


Рисунок 166 – Компонент DataSource в Delphi XE8

6.1. *ADO

ADO (ActiveX Data Objects – «объекты данных ActiveX») – интерфейс программирования приложений для доступа к данным, разработанный компанией Microsoft (MS Access, MS SQL Server) и основанный на технологии компонентов ActiveX. ADO позволяет представлять данные из разнообразных источников (реляционных баз данных, текстовых файлов и т. д.) в объектно-ориентированном виде.

В Delphi компоненты для работы с ADO расположены на вкладке «**ADO**» (рис. 167), а в новых версиях Delphi на вкладке «**dbGo**» (рис. 168).



Рисунок 167 – Компоненты ADO в Delphi 7

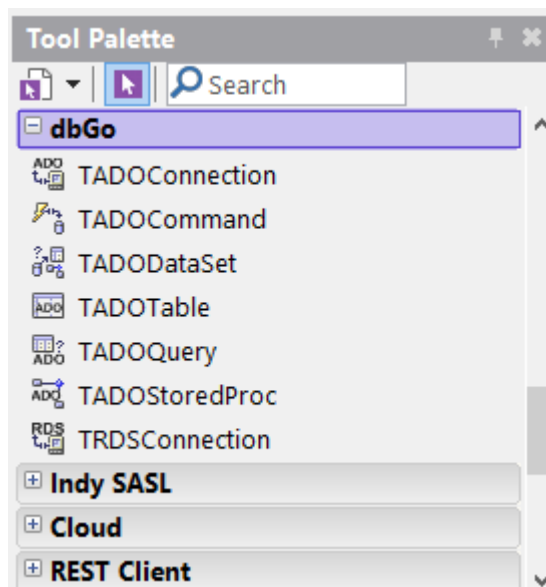


Рисунок 168 – Компоненты ADO в Delphi XE8

6.2. *Подключение к MS Access

Подготовка файла *.mdb

*По умолчанию Microsoft Access сохраняет базу данных в файл *.accdb, но рассматриваемый далее способ работает только с базами в формате *.mdb.*

В качестве примера возьмем базу данных, разработанную ранее в Работе № 1 (см. раздел 1.4). Данная база данных состояла из следующих объектов (рис. 169).

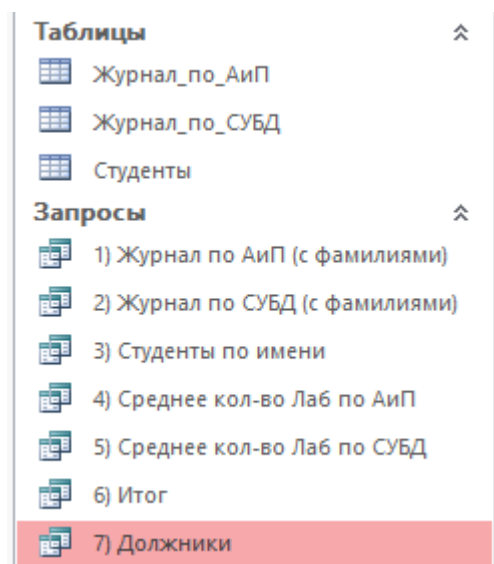


Рисунок 169 – Список объектов в Microsoft Access

Нажмем «Сохранить как» и пересохраним базу данных в формате ***.mdb**, например, с именем «**MyDatabase.mdb**».

!!! Этот файл необходимо разместить в той же папке, где будет располагаться проект Delphi.

Стоит обратить внимание, что база данных в этом формате не может содержать запросов (рисунке 170). Они были просто отброшены (в отличие от представленного на рисунке 169).

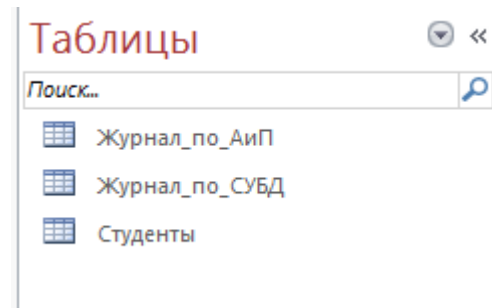


Рисунок 170 – Список объектов в файле *.mdb

Добавление компонентов в проект

Добавим в проект следующие 4 компонента:

- **ADOConnection** – подключение (соединение) с базой данных;
- **ADOQuery** – запрос;
- **DataSource** – источник данных;
- **DBGrid** – визуальный компонент для отображения таблицы.

После этого проект примет вид, представленный на рисунке 171.

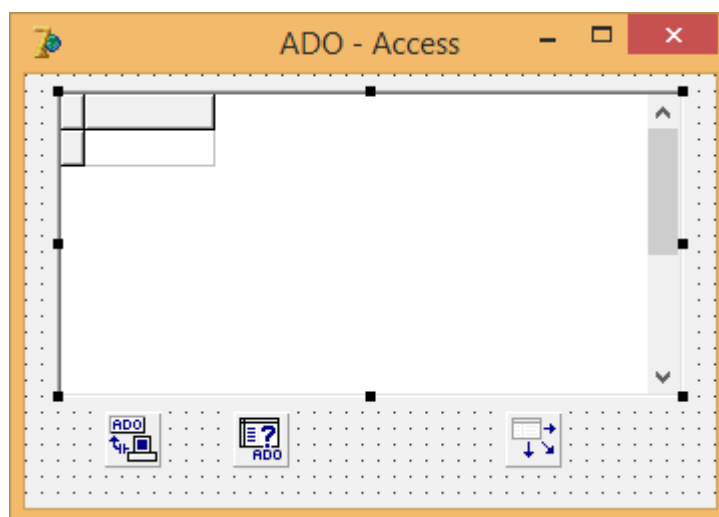


Рисунок 171 – Добавленные компоненты

Связь компонентов

Добавленные компоненты необходимо связать между собой следующим образом:

- связать **ADOQuery1.Connection** с **ADOConnection1** (рис. 172);

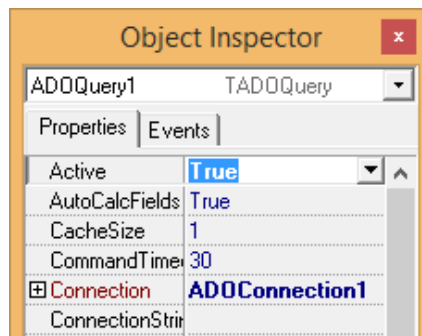


Рисунок 172 – Связь запроса с подключением

- связать **DataSource1.DataSet** с **ADOQuery1**;
- связать **DBGrid1.DataSource** с **DataSource1**;
- установить **ADOConnection1.LoginPrompt := False**.

В итоге мы получим следующую структуру в Дереве проекта (рис. 173).

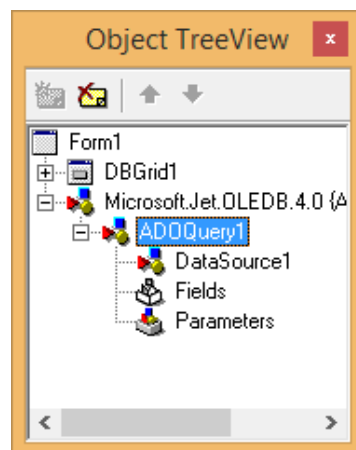


Рисунок 173 – Дерево проекта

Настройка соединения

Дважды кликнув мышкой по компоненту **ADOConnection**, попадаем в окно настройки соединения (рис. 174). В нем необходимо нажать кнопку «**Build**».

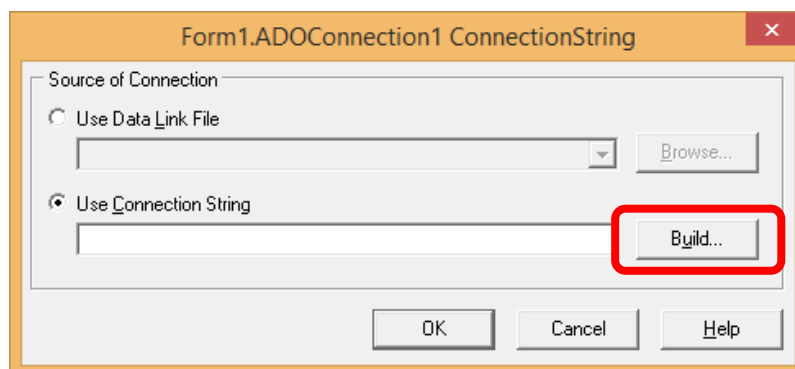


Рисунок 174 – Настройка соединения

Далее нам нужно выбрать поставщика данных. В нашем случае требуется «**Microsoft Jet 4.0 OLE DB Provider**» (рис. 175). Нажимаем кнопку «Далее».

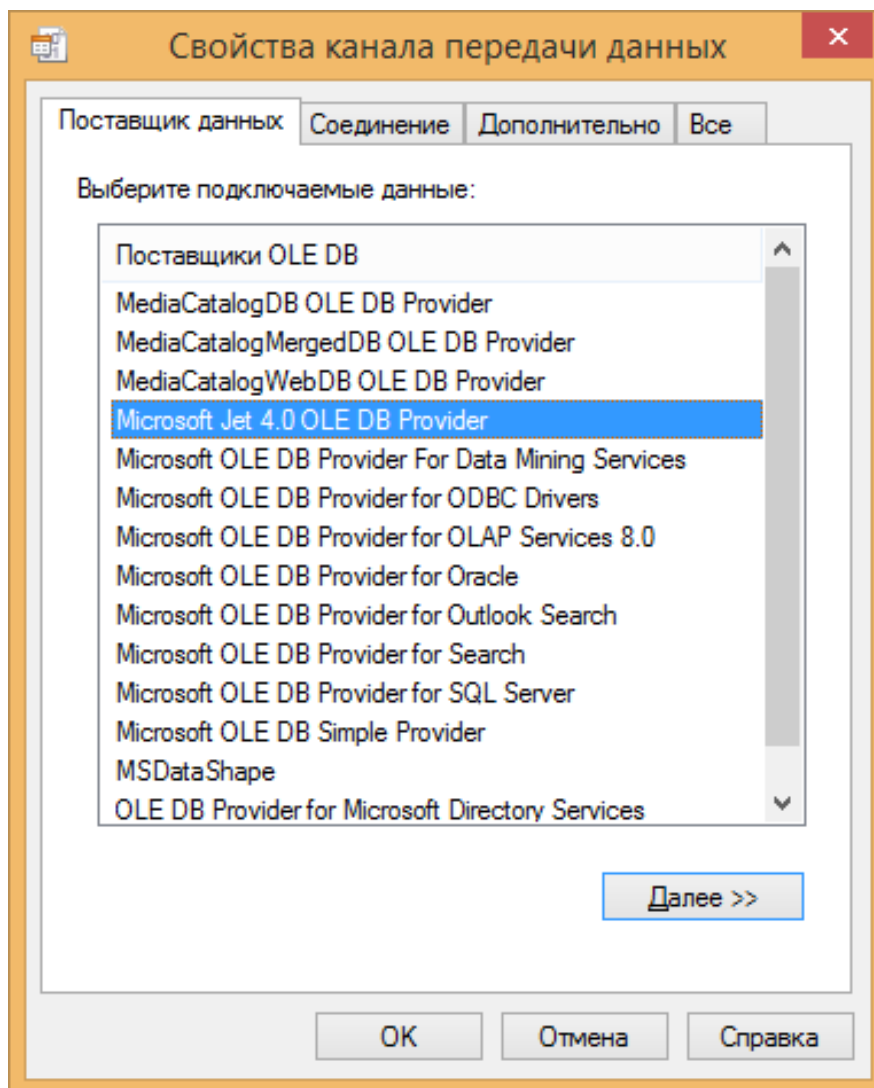


Рисунок 175 – Выбор поставщика данных

Теперь указываем имя файла с базой данных (рис. 176), которое также можно выбрать, нажав кнопку «...». Если этот файл лежит в той же папке, что и программа, то лучше указать **относительный** адрес (т.е. стереть путь к файлу). Это позволит перемещать и переименовывать папку с программой без необходимости перекомпилировать проект.

!!! Также имеет смысл сразу нажать кнопку «Проверить соединение» (см. рис. 176), чтобы убедиться, что все настроено правильно.

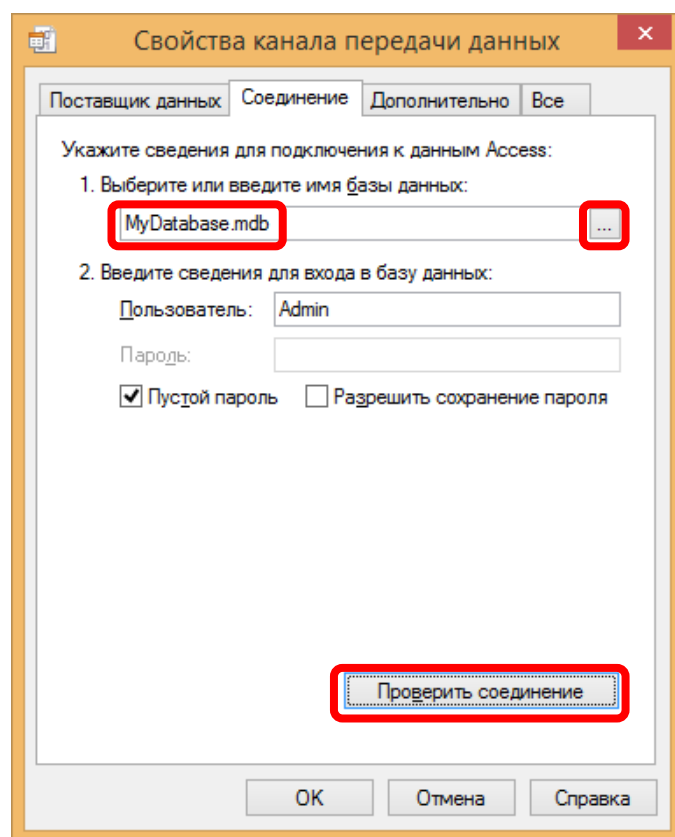


Рисунок 176 – Выбор файла

Подключение к базе данных

Теперь выберем компонент **ADOQuery** и дважды кликнем мышкой по его свойству «SQL» в Инспекторе объектов. В открывшемся окне введем SQL-запрос (рис. 177). Также здесь можно ввести любой из запросов, полученных ранее в *Работе № 1* (см. разделы **1.3 – 1.4**).

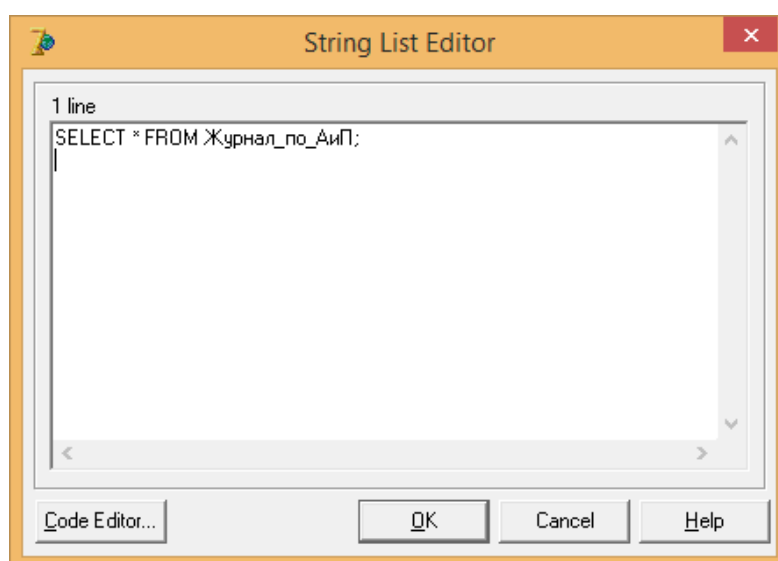
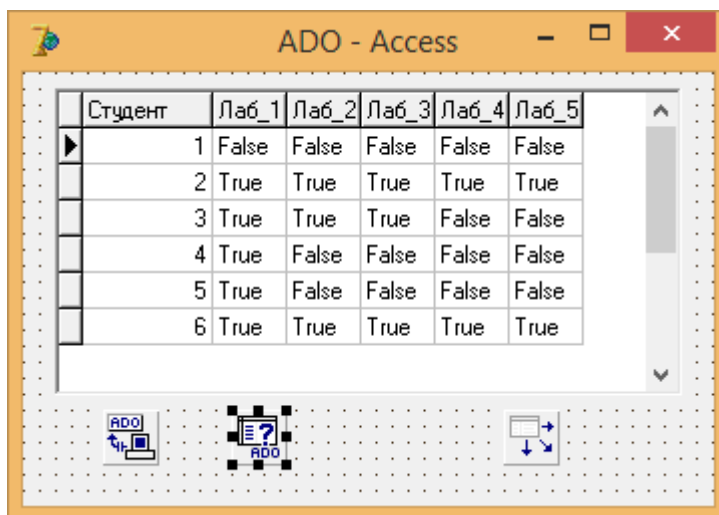


Рисунок 177 – SQL-запрос

После этого переведем свойство «**Active**» этого компонента в «**True**». В результате будет осуществлено подключение к базе данных, а запрошенные данные сразу отобразятся в **Delphi** (рис. 178).

!!! После каждого изменения текста SQL-запроса «**Active**» снова будет сбрасываться в «**False**», и его снова потребуется включить.



Студент	Лаб_1	Лаб_2	Лаб_3	Лаб_4	Лаб_5
1	False	False	False	False	False
2	True	True	True	True	True
3	True	True	True	False	False
4	True	False	False	False	False
5	True	False	False	False	False
6	True	True	True	True	True

Рисунок 178 – Результаты запроса

Также можно запускать подключение к базе, например, через кнопку. Для этого требуется написать код:

```
ADOQuery1.Active := True;
```

6.3. **ODBC

ODBC (Open Database Connectivity – «Открытый механизм взаимодействия с базами данных») – это программный интерфейс (**API**) доступа к базам данных, разработанный компанией Microsoft. Стандарт призван унифицировать программное взаимодействие с СУБД, сделать его независимым от поставщика СУБД и программно-аппаратной платформы.

ODBC обеспечивает независимость от СУБД, используя **драйвер ODBC** в качестве уровня трансляции между приложением и СУБД. Драйвер ODBC можно рассматривать как аналог драйвера принтера или другого устройства, предоставляющий стандартный набор функций для использования приложением и реализующий функциональность, специфичную для СУБД. Приложение, которое может использовать ODBC, называется «ODBC-совместимым». Любое ODBC-совместимое приложение может получить доступ к любой СУБД, для которой установлен драйвер. Драйверы существуют для всех основных СУБД и многих других источников данных, таких как Microsoft Excel, или даже для текстовых файлов или файлов с разделителями-запятыми (CSV).

6.4. *Подключение к Excel

Подготовка Excel- файла

Создадим в Excel таблицу с тремя столбцами, как показано на рисунке 179. У столбцов обязательно должны быть имена (в нашем примере это «x», «Sin(x)» и «Cos(x)»), т.к. именно эти имена будут использоваться дальше в SQL-запросах. *Также можно построить график (но он дальше использоваться не будет).*

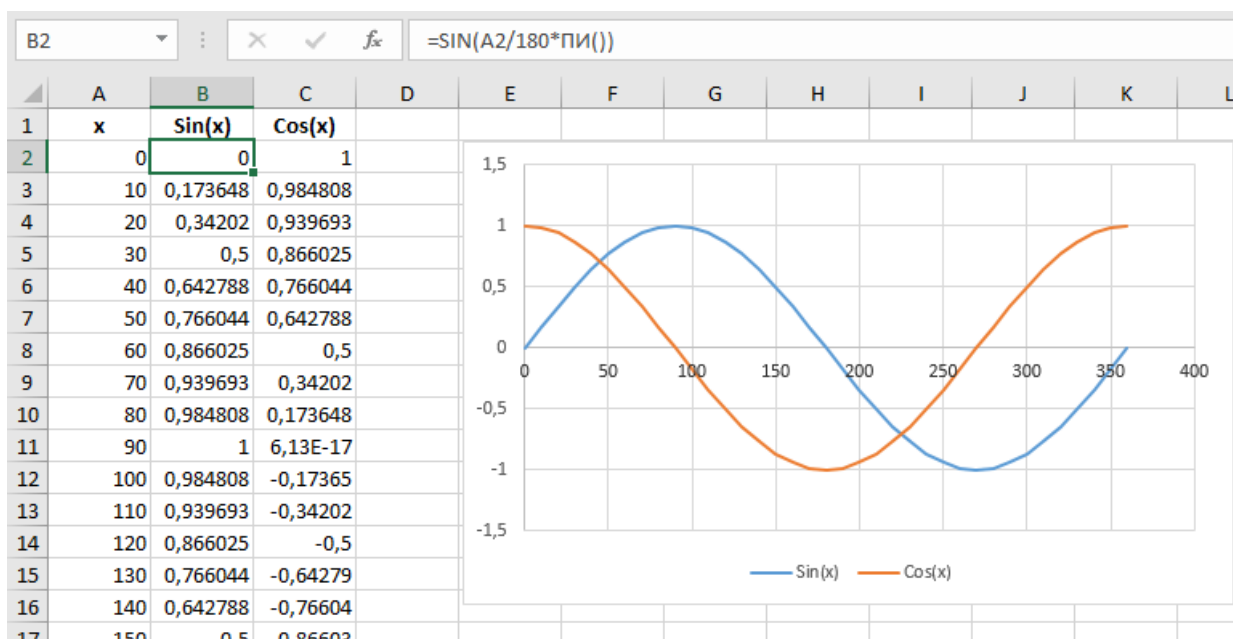


Рисунок 179 – Таблица в Excel

Далее необходимо воспользоваться функцией «**Сохранить как**» и **обязательно** сохранить таблицу в файл с расширением ***.xls** (а не *.xlsx). В нашем случае назовем файл «**SinCos.xls**».

Создание проекта Delphi

Добавим в проект следующие компоненты:

- **ADOConnection** (установим для него **LoginPrompt := False**);
- **ADOQuery** (свяжем его с **ADOConnection**);
- **DataSource** (свяжем его с **ADOQuery**);
- **DBGrid** (свяжем его с **DataSource**).

После этого проект примет вид, представленный на рисунке 180.

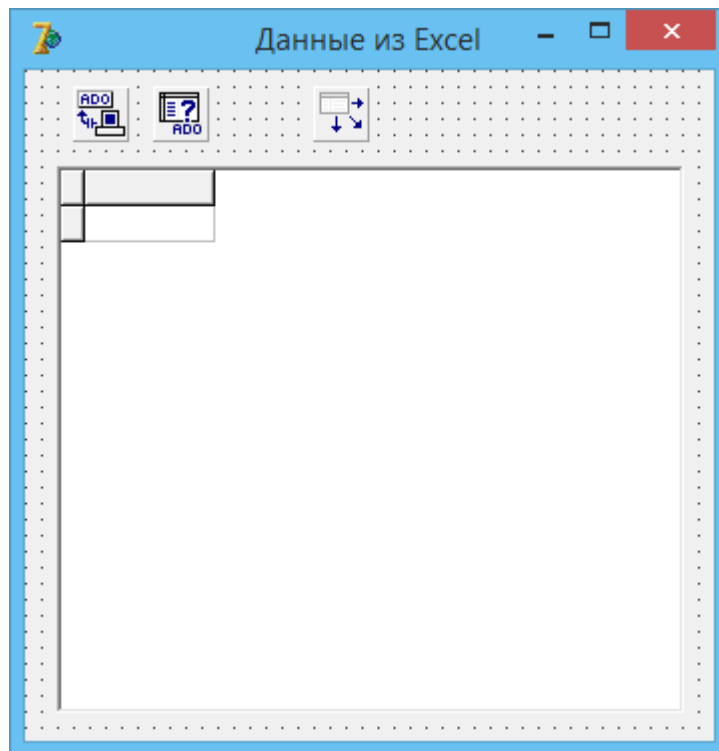


Рисунок 180 – Добавленные компоненты

Настройка соединения

Дважды кликнув мышкой по компоненту **ADOConnection** и в открывшемся окне (рис. 181) нажмем кнопку «**Build**».

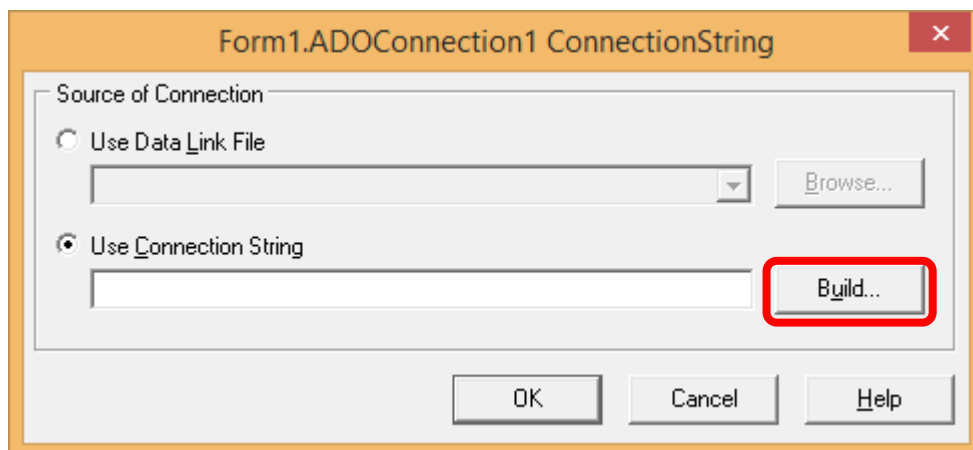


Рисунок 181 – Настройка соединения

В следующем окне выбираем поставщика «**Microsoft OLE DB Provider for ODBC Drivers**» (рис. 182) и нажимаем кнопку «**Далее**».

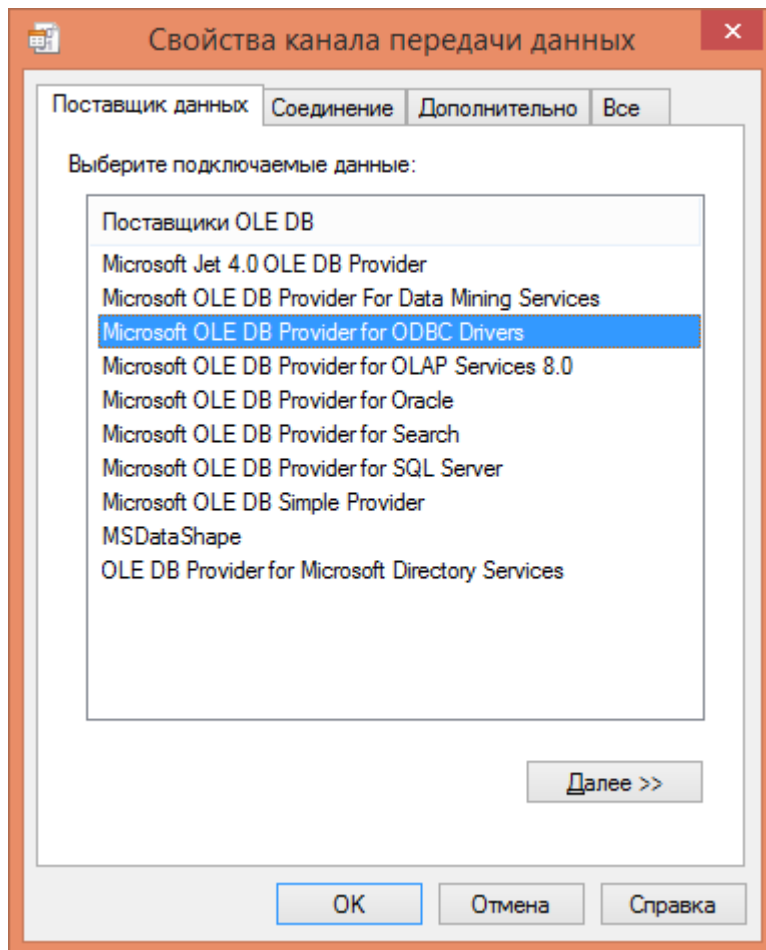


Рисунок 182 – Выбор поставщика данных

Теперь необходимо выбрать вариант «Использовать строку соединения» и нажать кнопку «Сборка» (рис. 183).

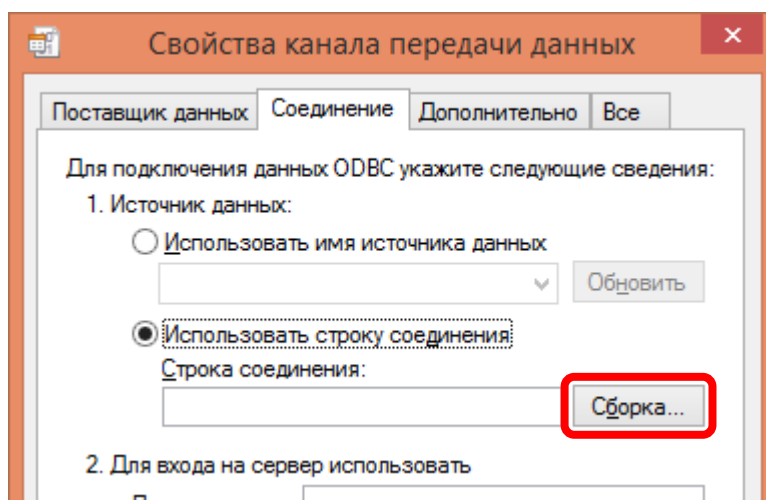


Рисунок 183 – Настройка соединения

Далее необходимо перейти на вторую вкладку (рис. 184) и дважды кликнуть по источнику данных «Файлы Excel».

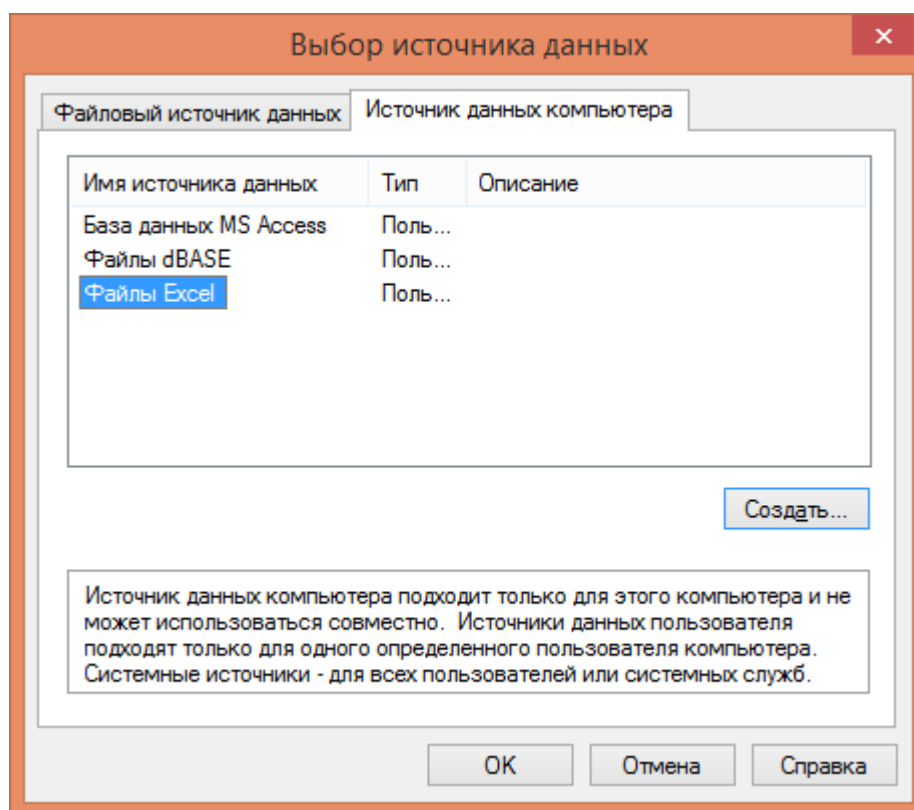


Рисунок 184 – Выбор источника данных

В открывшемся окне выбираем созданный ранее файл «**SinCos.xls**» (рис. 185).

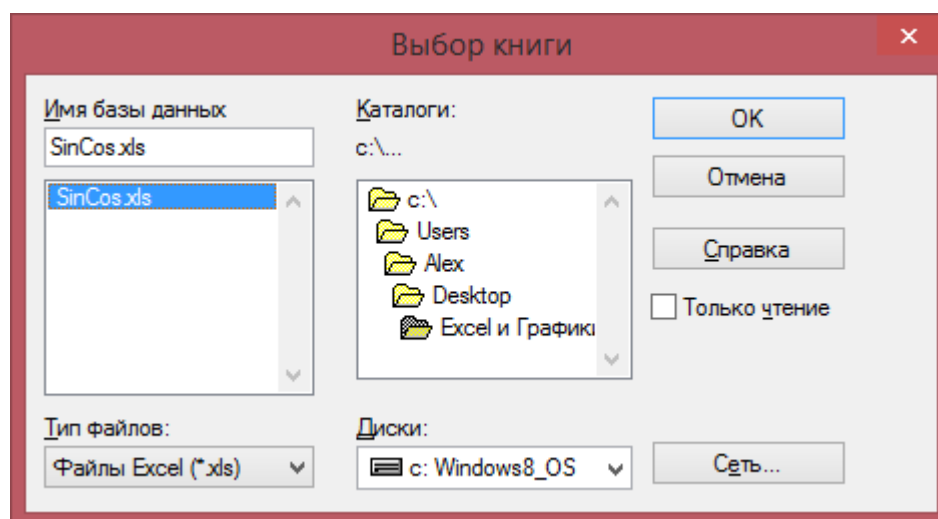


Рисунок 185 – Выбор файла с базой данных

В результате строка подключения (рис. 186) примет следующий вид:
 Provider=MSDASQL.1;Persist Security Info=False;Extended Properties="DSN=Файлы Excel;DBQ=C:\Users\Alex\Desktop\Excel и Графики\SinCos.xls;DefaultDir=C:\Users\Alex\Desktop\Excel и Графики;DriverId=790;MaxBufferSize=2048;PageTimeout=5;"

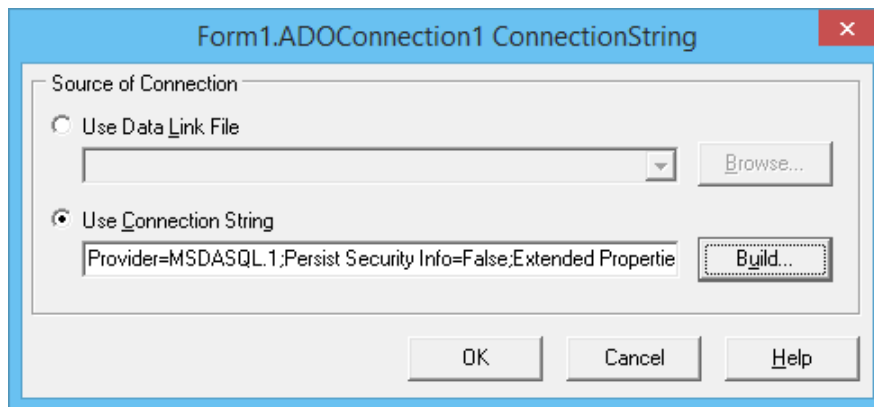


Рисунок 186 – Строка подключения

SQL-запрос

Пропишем в компоненте **ADOQuery** запрос, представленный на рисунке 187. Здесь вместо имени таблицы используется имя Листа в Excel. Причем это имя листа должно быть заключено в квадратные скобки, а после него должен следовать символ «\$».

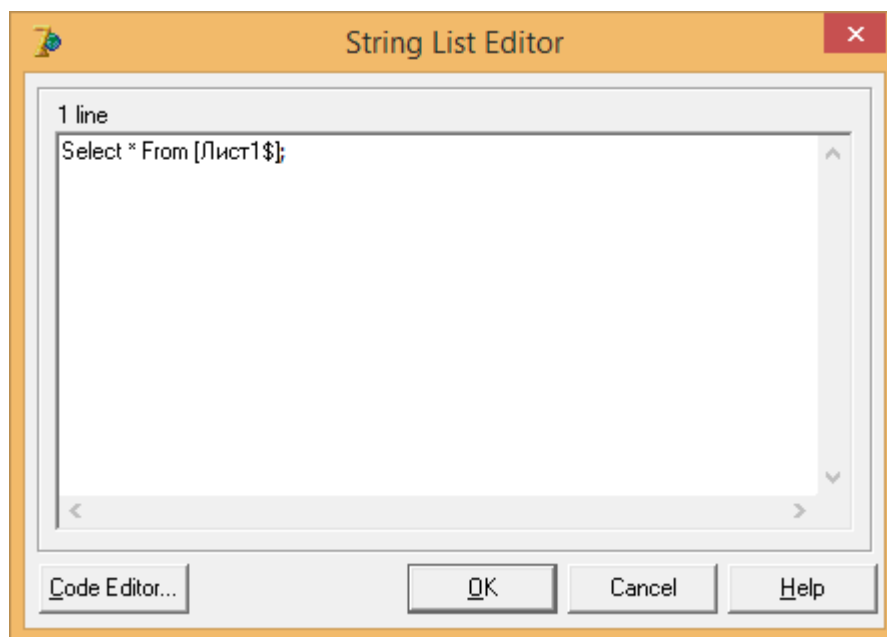


Рисунок 187 – SQL-запрос

Подключение к базе будет происходить при старте программы в **OnCreate** формы. Для этого напишем следующий код:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    ADOQuery1.Active := True;
end;
```

Запустим программу и получим данные из Excel-файла (рис. 188).

x	Sin(x)	Cos(x)
0	0	1
10	0,17364817766693	0,984807753012208
20	0,342020143325669	0,939692620785908
30	0,5	0,866025403784439
40	0,642787609686539	0,766044443118978
50	0,766044443118978	0,642787609686539
60	0,866025403784439	0,5
70	0,939692620785908	0,342020143325669
80	0,984807753012208	0,17364817766693
90	1	0,1257422745431E-17
100	0,984807753012208	-0,17364817766693
110	0,939692620785908	-0,342020143325669
120	0,866025403784439	-0,5
130	0,766044443118978	-0,642787609686539

Рисунок 188 – Итоговая программа

****Усовершенствование работы**

Есть одна проблема: если мы хотим, чтобы база данных (в виде Excel-файла) располагалась в одной папке с программой и мы могли перемещать эту папку в произвольное место или переименовывать ее – то этот способ не подходит. Т.к. путь к базе данных прописывается жестко, и файл с базой данных нельзя перемещать (либо программу каждый раз придется перекомпилировать).

Для решения этой проблемы необходимо на этапе формирования строки подключения (см. рис. 186) скопировать эту строку в буфер обмена, а соответствующее поле очистить (т.е. использовать для подключения **пустую** строку).

Далее используем скопированную строку в коде:

```

procedure TForm1.FormCreate(Sender: TObject);
var CD: String;
begin
    //Текущая папка
    CD := ExtractFilePath(ParamStr(0));
    //Подключение к БД
    ADOConnection1.ConnectionString := 'Provider=MSDASQL.1;' +
        'Persist Security Info=False;' +
        'Extended Properties="DSN=файлы Excel;' +
        'DBQ=' + CD + 'SinCos.xls;' +
        'DefaultDir=' + CD + ';" +
        'DriverId=790;' +
        'MaxBufferSize=2048;' +
        'PageTimeout=5;';

    ADOQuery1.Active := True;
end;

```

Здесь при запуске программы мы подставляем в строку подключения адрес текущей папки («CD»), из которой была запущена программа.

6.5. *Отображение графиков

Добавление графика

Добавим к предыдущему примеру (см. раздел 6.4) график **TDBChart** (рис. 189).

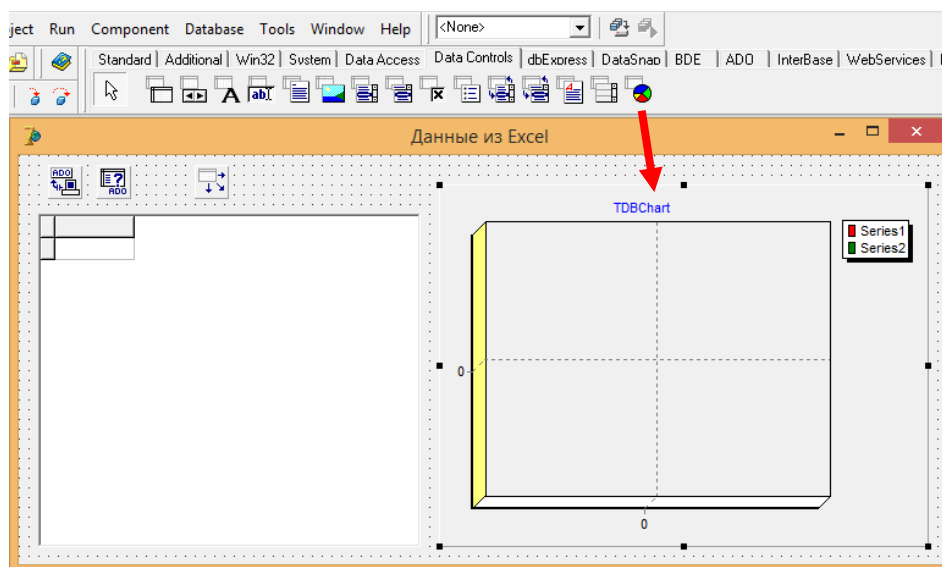


Рисунок 189 – График TDBChart

Далее дважды кликнем по графику и добавим («Add») две серии (рис. 190). Для каждой серии появится окно с выбором типа графика (рис. 191).

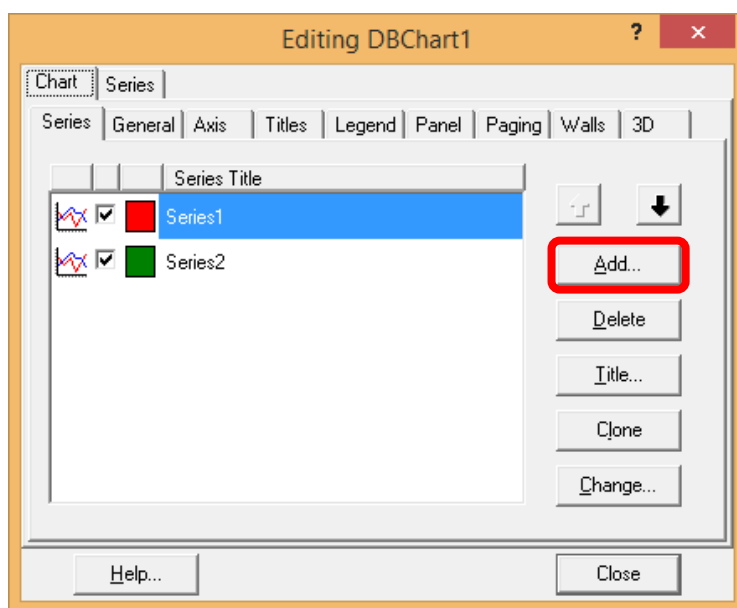


Рисунок 190 – Добавление серий данных

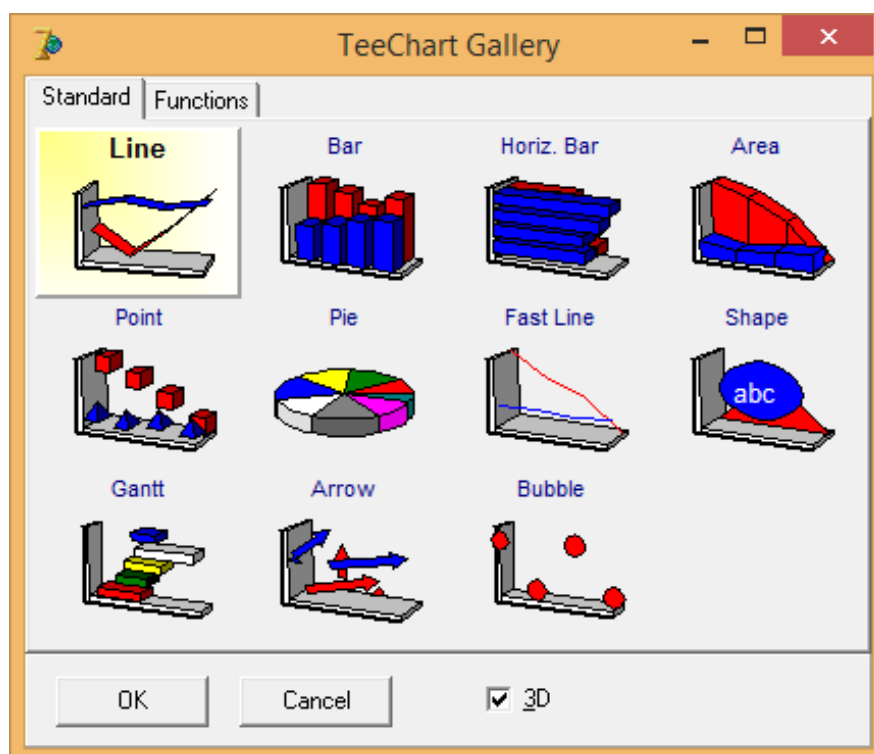


Рисунок 191 – Выбор типа графика

Настройка графика

Каждую из добавленных Серий нужно связать с нашим SQL-запросом (рис. 192).

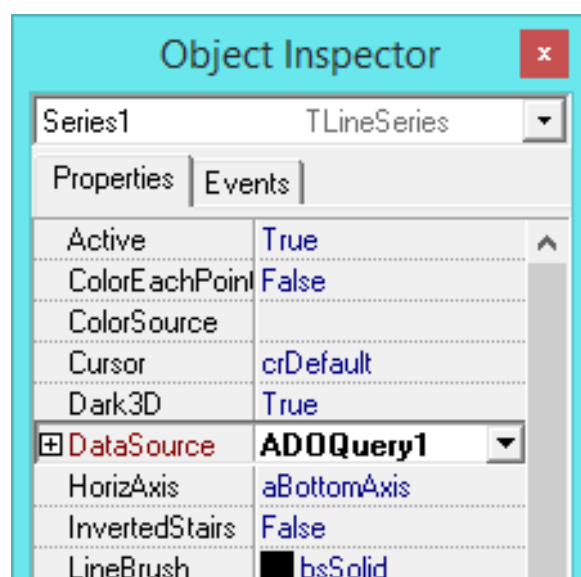


Рисунок 192 – Связь серии с SQL-запросом

Далее выбираем из Excel-таблицы столбцы, которые будут отображаться (рис. 193).

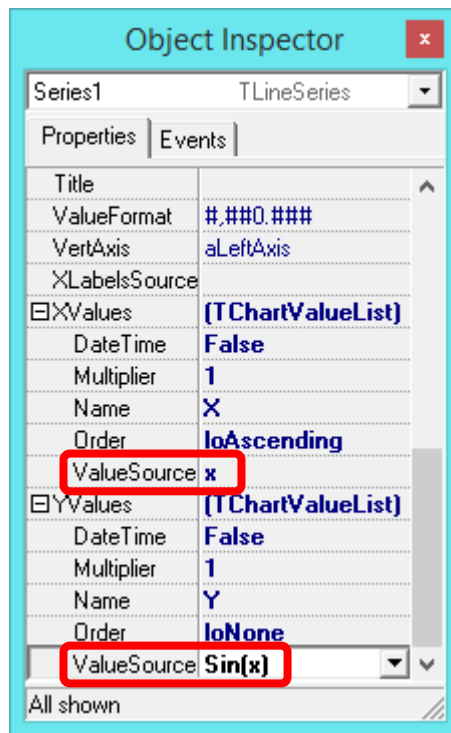


Рисунок 193 – Выбор столбцов

Внешний вид итоговой программы представлен на рисунке 194.

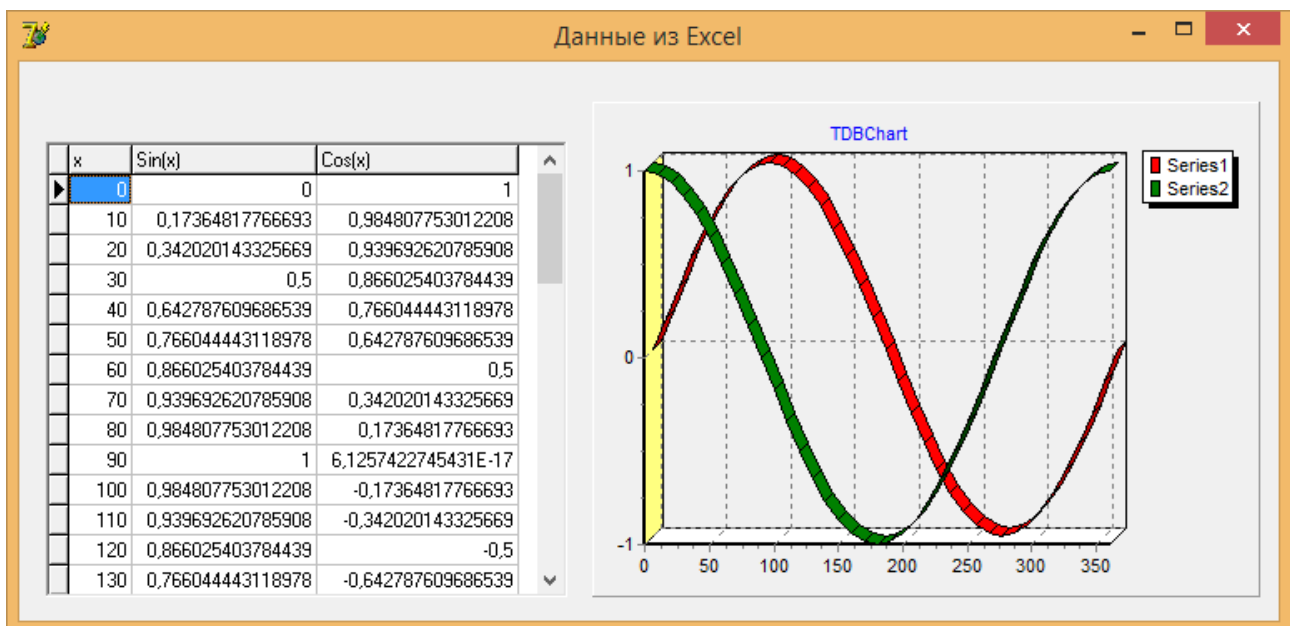


Рисунок 194 – Итоговая программа с графиком

6.6. **Добавление нового источника данных

Если нам недостаточно источников данных в предлагаемом списке (рис. 195), то необходимо нажать кнопку «Создать».

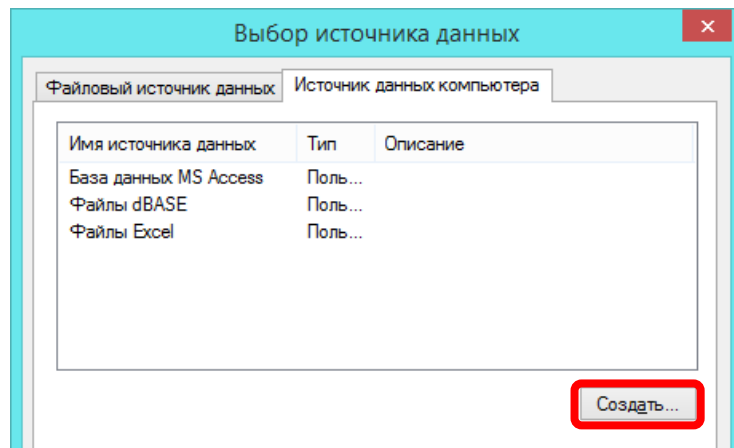


Рисунок 195 – Создание источника данных

Скорее всего появится предупреждение (рис. 196), а в следующем окне (рис. 197) нам не будет доступен второй случай. Если нам необходим «системный» вариант, то необходимо перезапустить всю среду **Delphi** с права **Администратора**, но обычно в этом нет необходимости.

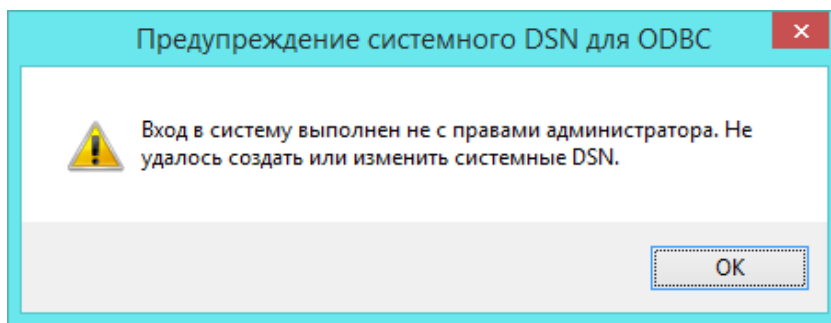


Рисунок 196 – Предупреждение об отсутствии прав

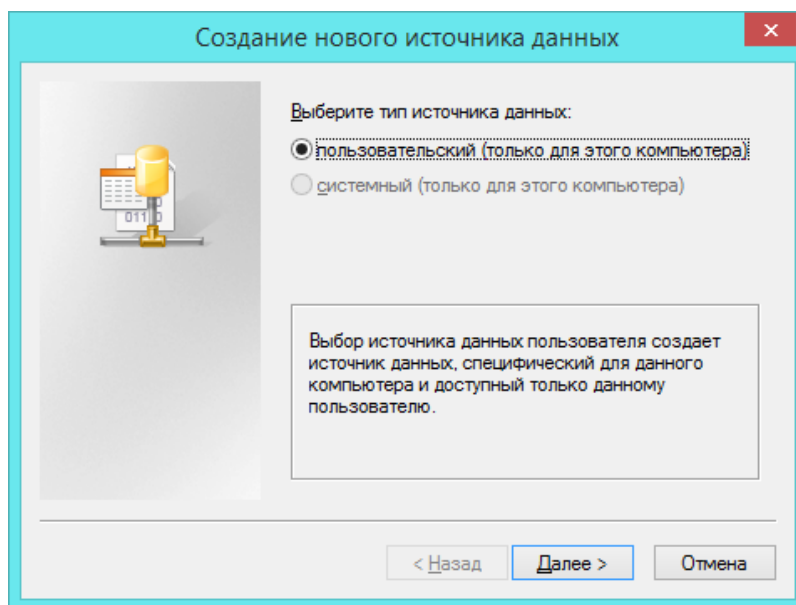


Рисунок 197 – Выбор типа источника

Теперь мы видим полный список вариантов (рис. 198), которые мы можем добавить.

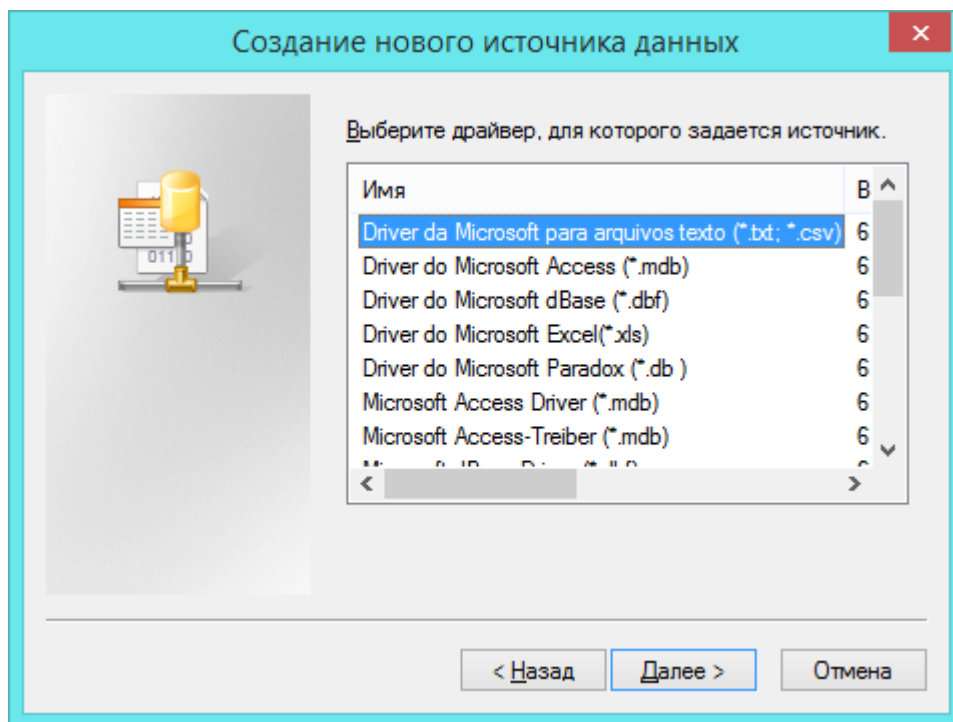


Рисунок 198 – Варианты источников данных

При добавлении нового источника нам необходимо ввести для него имя (рис. 199) и провести другие настройки. После этого новый источник данных отобразится в списке (рис. 200).

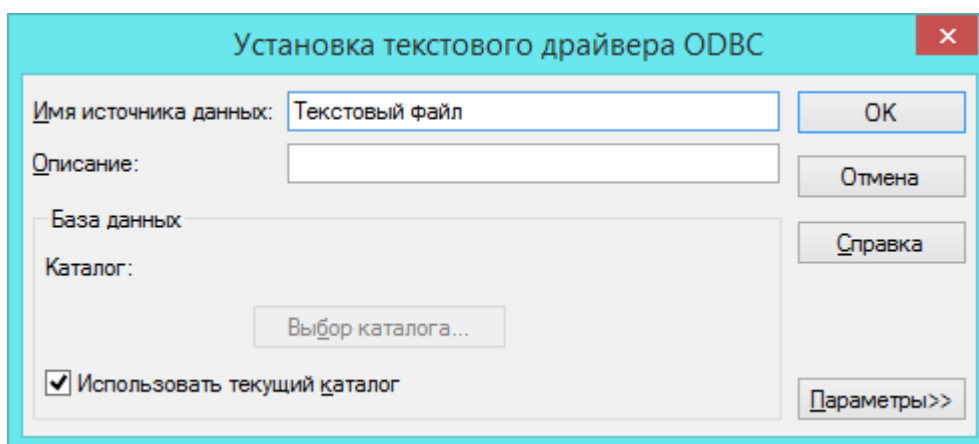


Рисунок 199 – Настройка нового источника

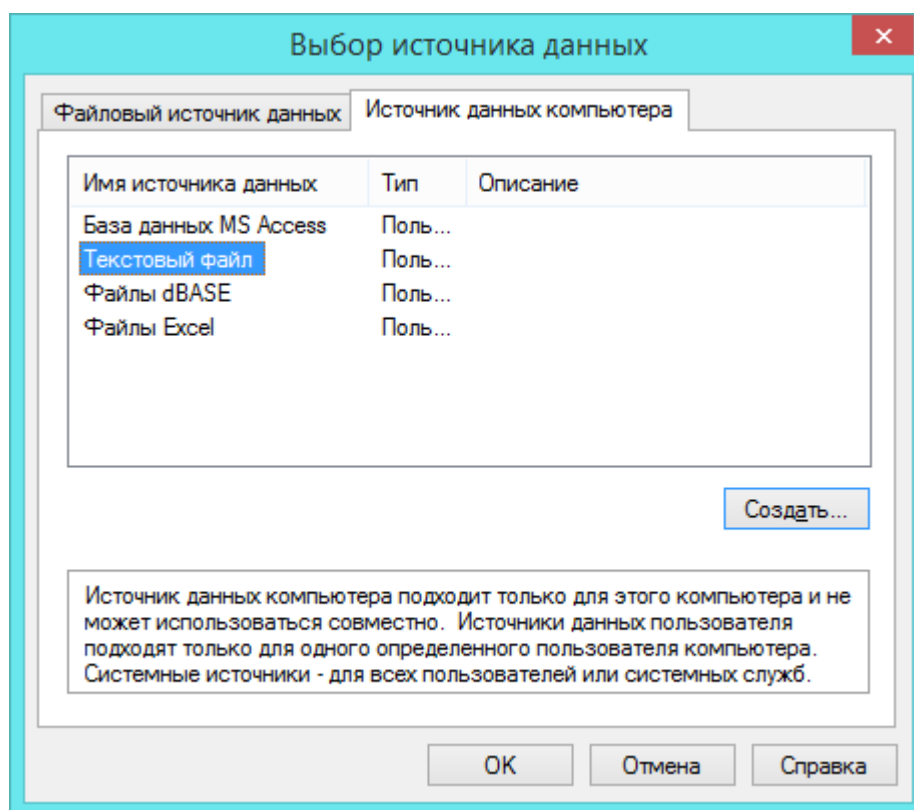


Рисунок 200 – Добавленный источник данных

Если, как и раньше, дважды кликнуть по этому источнику, то мы сможем выбрать файл с базой данных указанного типа (рис. 201).

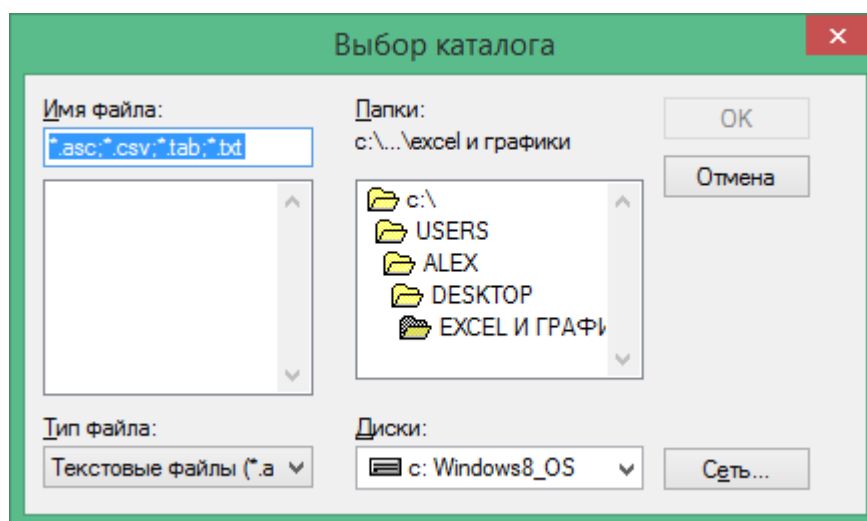


Рисунок 201 – Выбор текстового файла с базой данных

Удалить добавленные источники данных из списка не получится, это можно сделать только из Панели управления Windows (см. далее раздел 6.7).

6.7. **Администрирование ODBC в Windows

Данный раздел не имеет прямого отношения к Delphi. Как и многие другие программы, Delphi только использует данные настройки системы для своей работы.

Откроем «Панель управления» Windows (рис. 202) и в строке поиска введем «ODBC» (рис. 203).

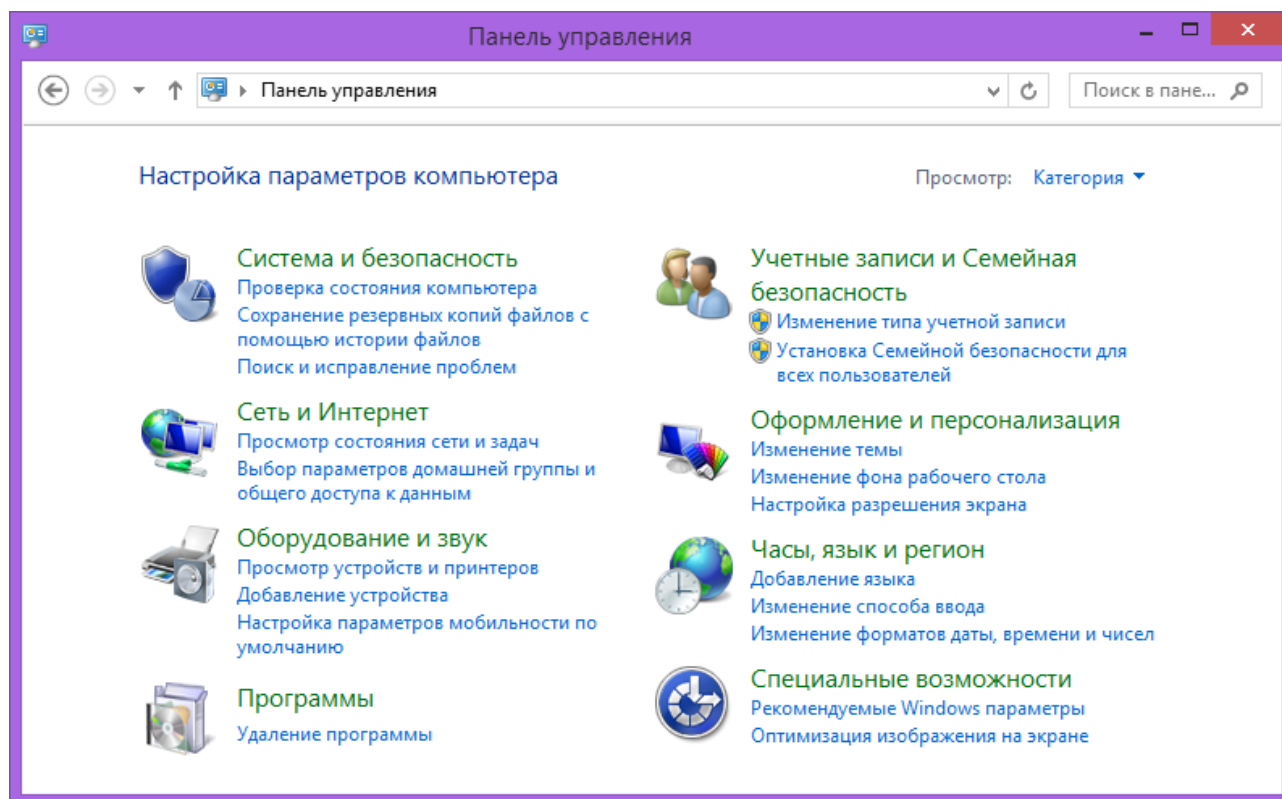


Рисунок 202 – Панель управления

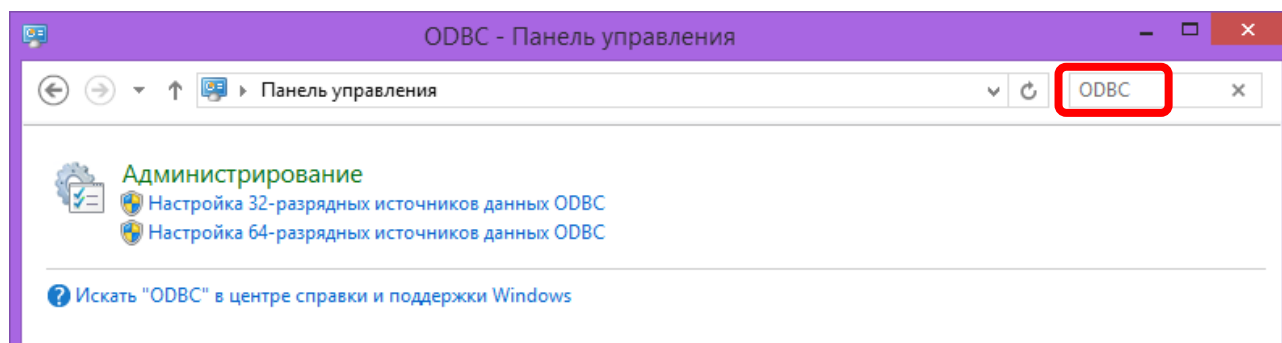


Рисунок 203 – Источники данных ODBC

Выберем из списка 32-разрядные источники данных и в списке увидим те самые источники (рис. 204), которые ранее мы видели в Delphi. Здесь можно удалить источники, созданные в Delphi.

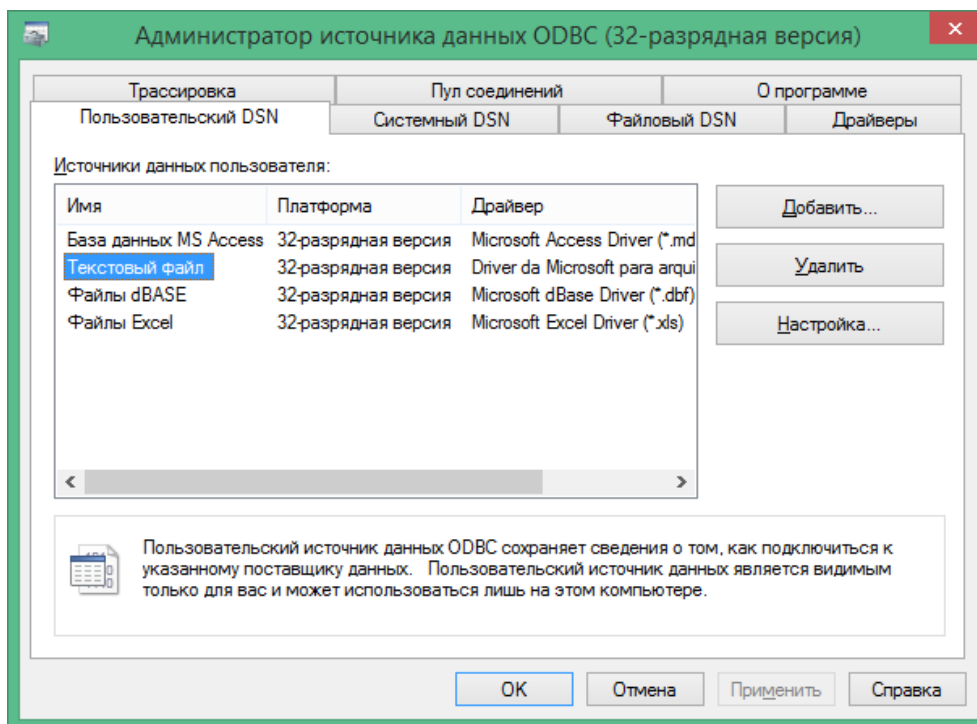


Рисунок 204 – Список источников данных в Windows

На вкладке «Драйверы» (рис. 205) мы можем увидеть список установленных в Windows драйверов ODBC, на основе которых мы можем добавлять источники данных.

При желании можно устанавливать и новые драйверы ODBC для работы с другими поставщиками услуг. Другие драйверы можно скачать в Интернете.

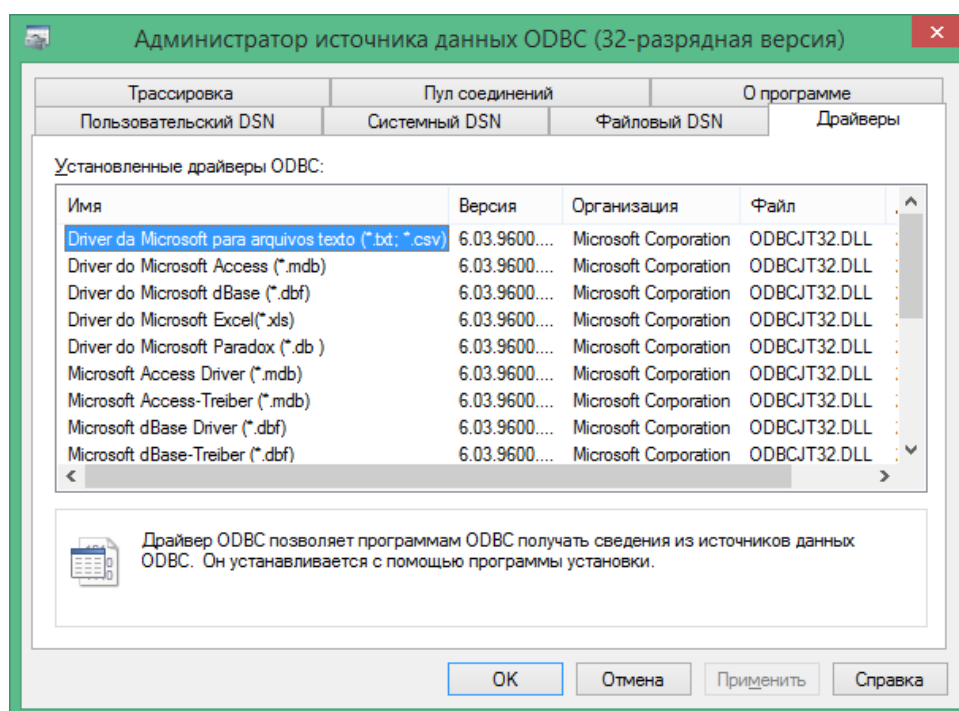


Рисунок 205 – Драйверы ODBC

6.8. **Подключение к MySQL

Скачивание и установка драйвера

Драйвер (Коннектор) для подключения к базе данных MySQL можно скачать по адресу: <https://dev.mysql.com/downloads/connector/odbc/>.

Нас интересует **32-битная версия** (*Delphi 7 не сможет работать с 64-битным драйвером, а новые версии Delphi смогут работать с ним, только если мы компилируем наш проект как 64-битный*).

Скачиваем установщик (*.msi) для Windows (рис. 206). Также на сайте имеются установочные пакеты для macOS (*.dmg) и для Linux (*.deb).

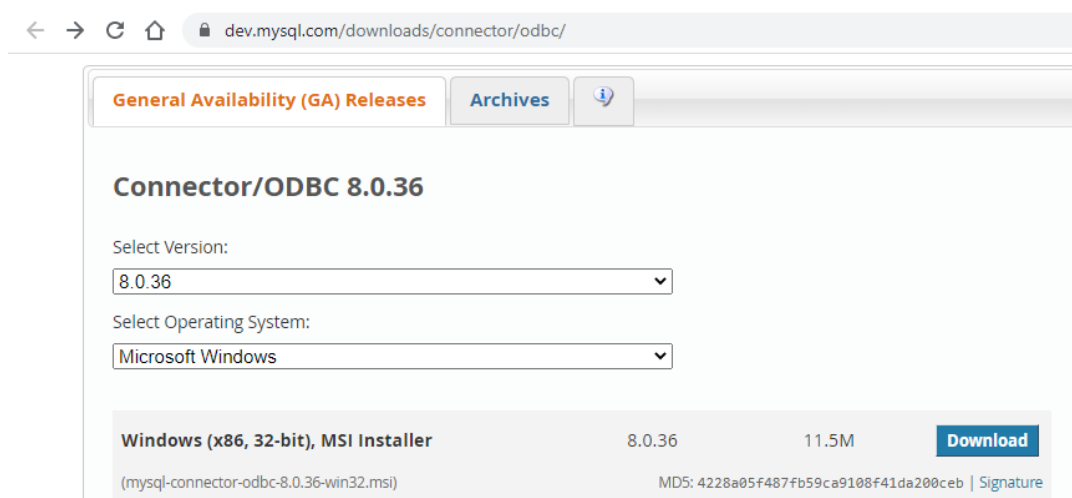


Рисунок 206 – Скачивание драйвера

Создание проекта Delphi

Добавим в проект следующие компоненты:

- **ADOConnection** (установим для него **LoginPrompt := False**);
- **ADOQuery** (свяжем его с **ADOConnection**);
- **DataSource** (свяжем его с **ADOQuery**);
- **DBGrid** (свяжем его с **DataSource**).

После этого проект примет вид, представленный на рисунке 207.

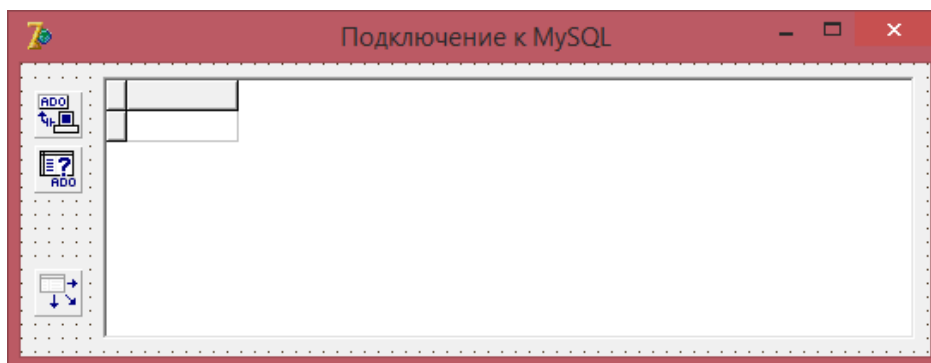


Рисунок 207 – Добавленные компоненты

Настройка соединения

Дважды кликнув мышкой по компоненту **ADOConnection** и в открывшемся окне нажмем кнопку **«Build»**.

В следующем окне выбираем поставщика **«Microsoft OLE DB Provider for ODBC Drivers»** (рис. 208) и нажимаем кнопку **«Далее»**.

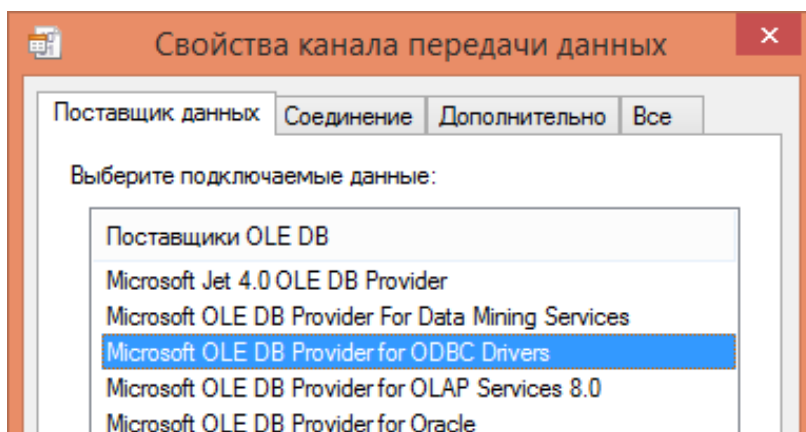


Рисунок 208 – Выбор поставщика данных

После этого необходимо выбрать вариант **«Использовать строку соединения»** и нажать кнопку **«Сборка»** (рис. 209).

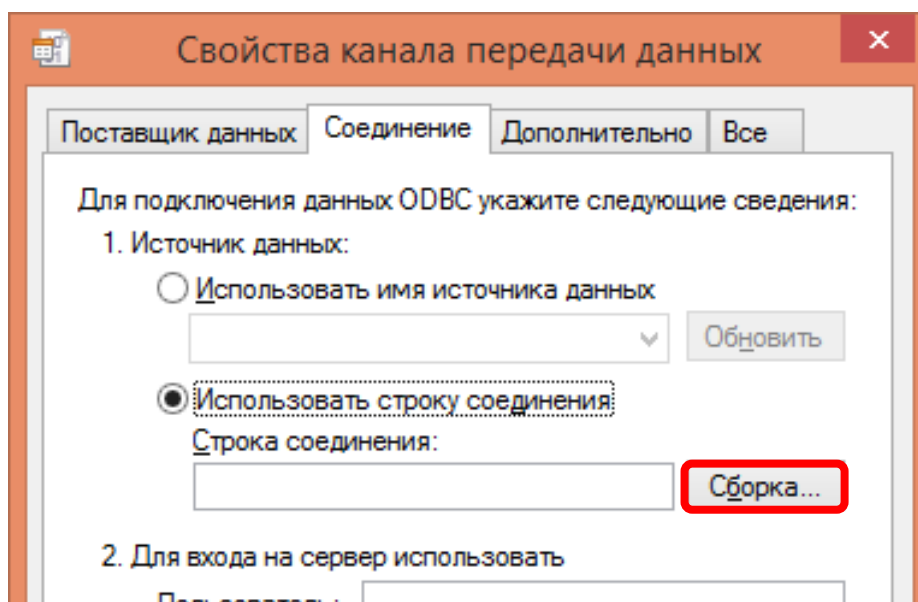


Рисунок 209 – Настройка соединения

Нажимаем кнопку **«Создать»** (рис. 210) и выбираем драйвер **«MySQL ODBC»** (рис. 211).

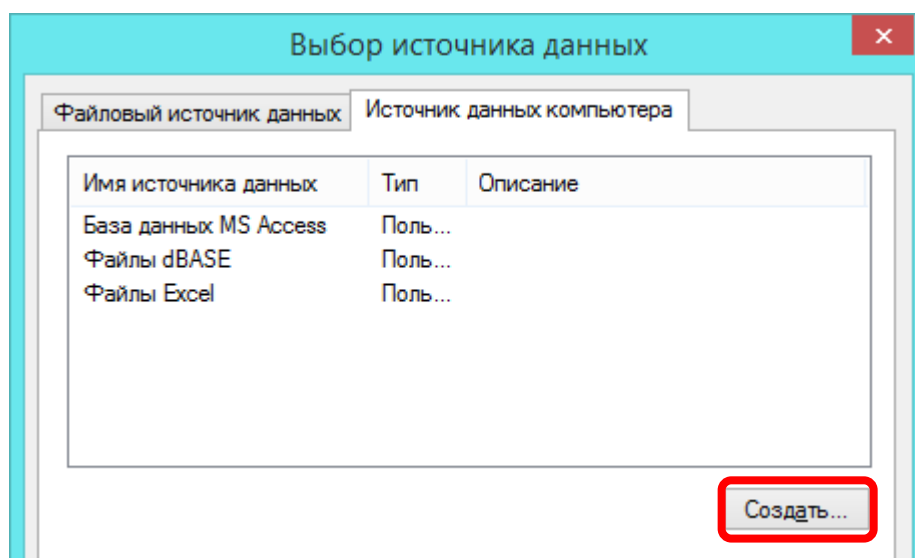


Рисунок 210 – Создание источника данных

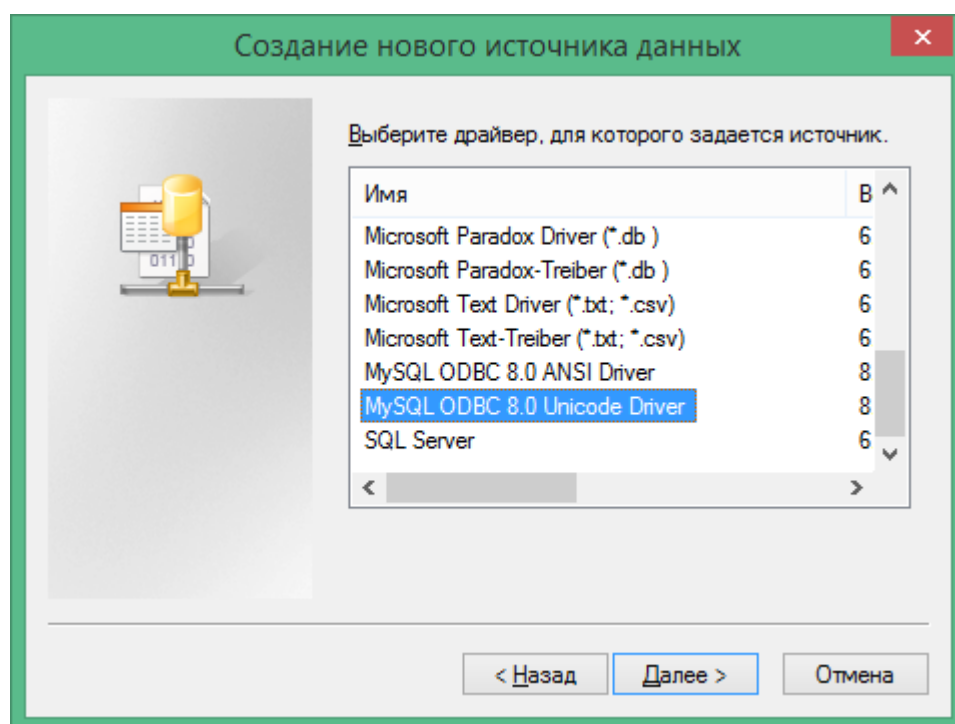


Рисунок 211 – Выбор драйвера MySQL

Теперь необходимо указать имя соединения, адрес сервера и имя пользователя (рис. 212).

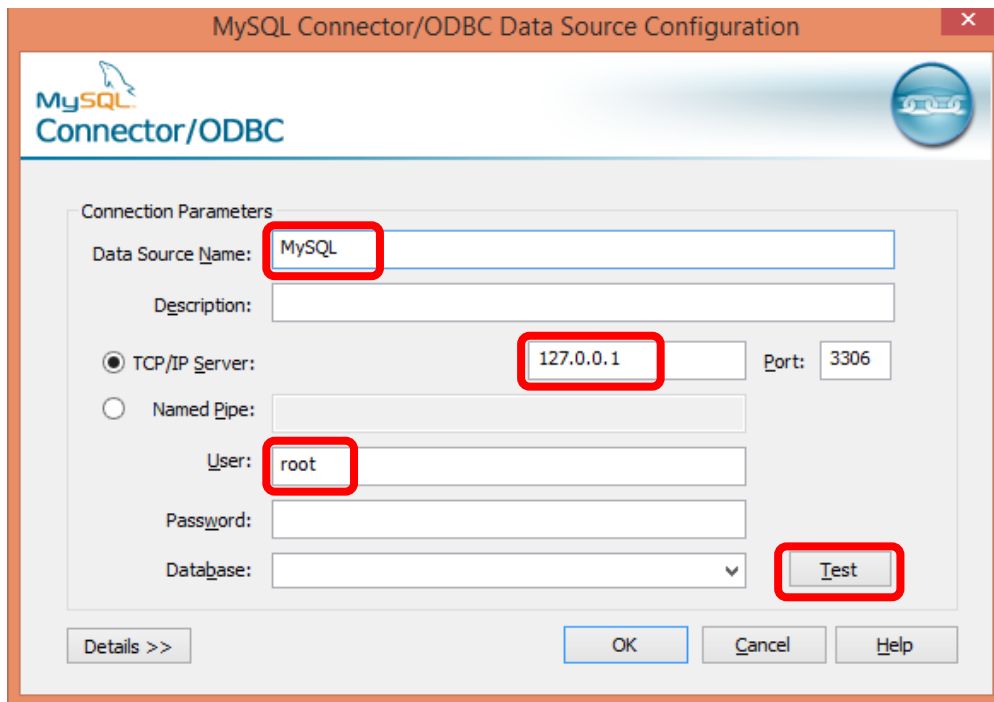


Рисунок 212 – Настройка подключения к серверу MySQL

Нажмем кнопку «**Test**» для проверки соединения. В случае успешного подключения отобразится окно, представленное на рисунке 213. В случае ошибки подключения мы увидим окно, представленное на рисунке 214.

!!! WAMP-сервер в этот момент уже должен быть запущен, а его значок в трее (у часов) должен быть зеленым.

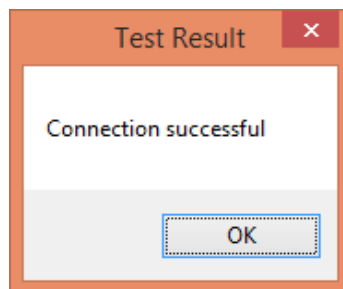


Рисунок 213 – Успешное подключение

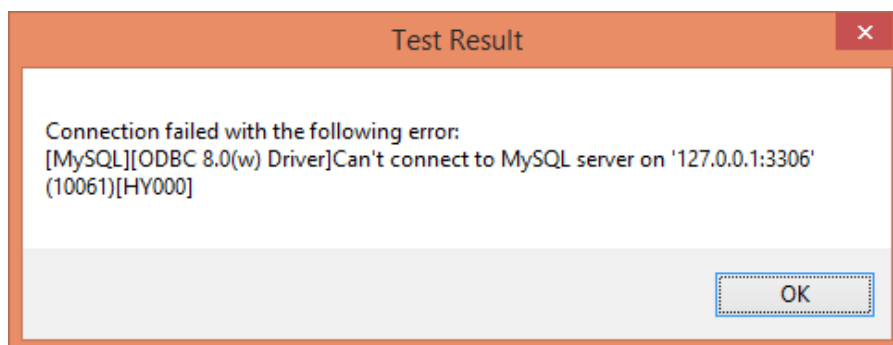


Рисунок 214 – Ошибка подключения

Осталось только выбрать базу, к которой необходимо подключиться (рис. 215). Ранее мы создавали базу, имя которой содержало фамилию автора работы.

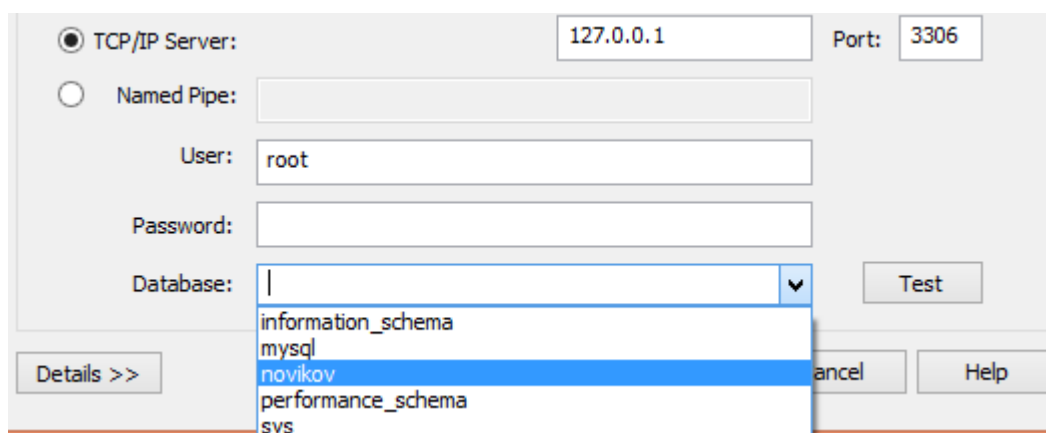


Рисунок 215 – Выбор базы данных

В результате в список источников данных будет добавлен «MySQL» (рис. 216).

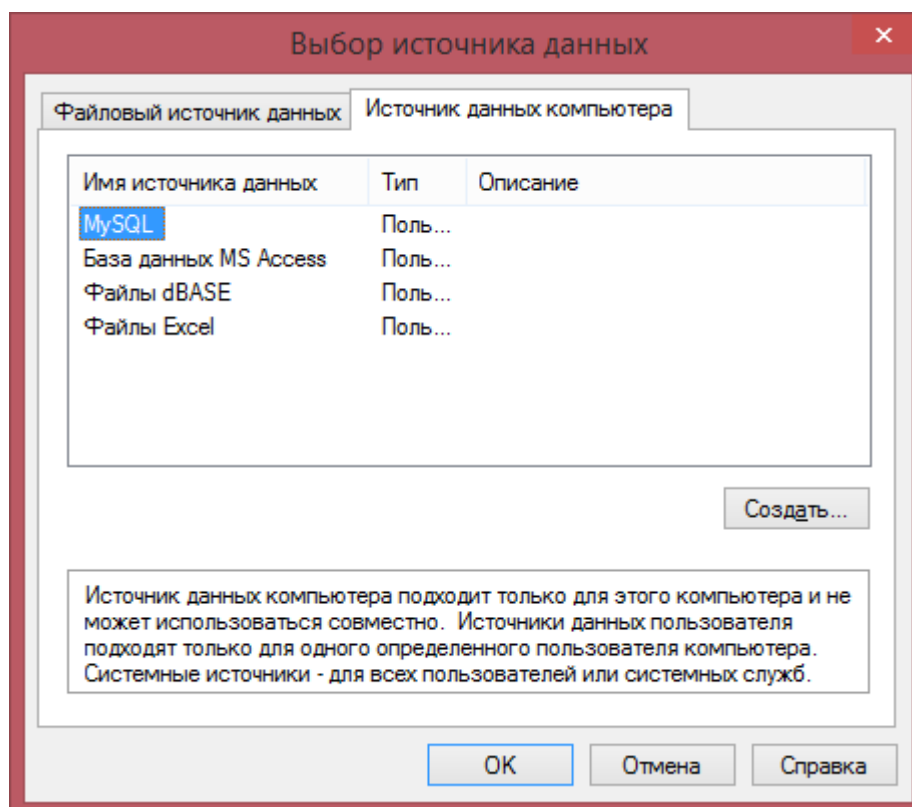


Рисунок 216 – Добавленный источник данных

SQL-запрос

Пропишем в компоненте **ADOQuery** запрос, представленный на рисунке 217. Мы будем получать данные из таблицы **test3_tbl**, которая была рассмотрена ранее в разделе 5.2 (см. рис. 144).

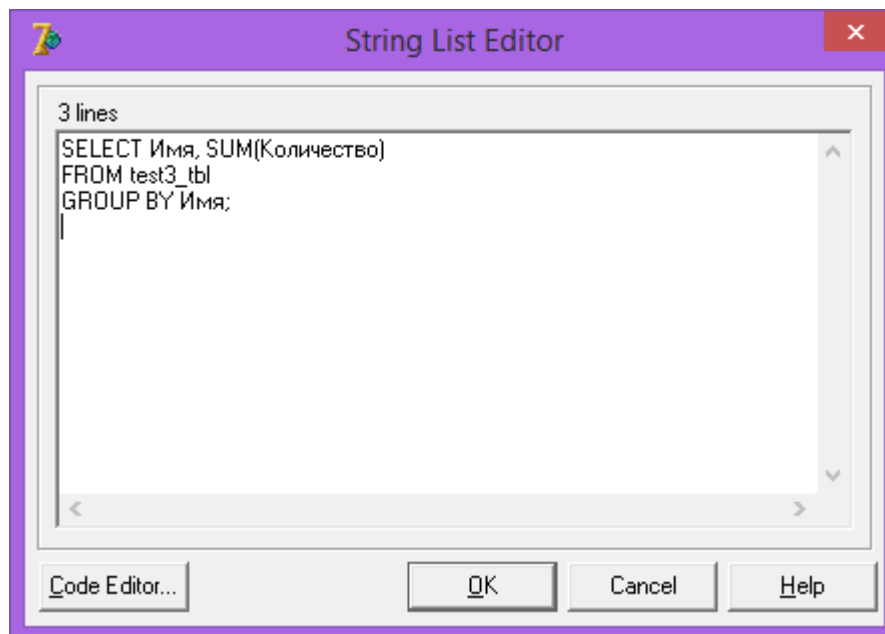


Рисунок 217 – SQL-запрос

Подключение к базе будет происходить при старте программы, в **OnCreate** формы. Для этого напишем следующий код:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    ADOQuery1.Active := True;
end;
```

Запустим программу и получим данные с сервера MySQL (рис. 218).

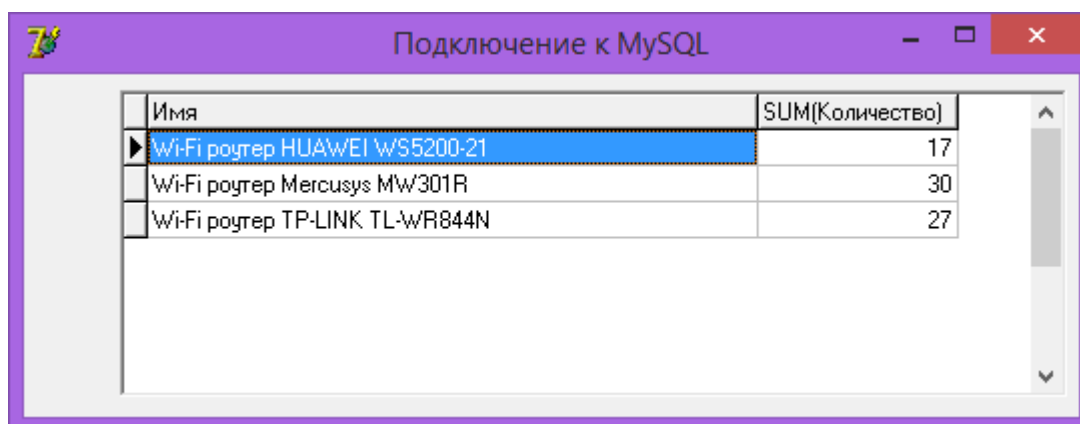


Рисунок 218 – Итоговая программа

7. *САМОСТОЯТЕЛЬНОЕ УГЛУБЛЕННОЕ ИЗУЧЕНИЕ

В данной главе представлена информация, которая будет полезна студенту для самостоятельного углубленного изучения работы **WAMP**-сервера, **Apache**-сервера, **MySQL**-сервера, языка **PHP** и их настройки.

7.1. *Настройка прав доступа к MySQL

По умолчанию любой может получить доступ к **MySQL** через **PhpMyAdmin**, поэтому необходимо установить пароль на учетную запись «**root**». А также требуется разграничить права доступа для разных пользователей и разных баз данных.

1. Пароль учетной записи «root»

Для настройки пароля администратора (учетная запись «**root**») необходимо нажать надпись «Сервер: MySQL» вверху страницы (рис. 219).

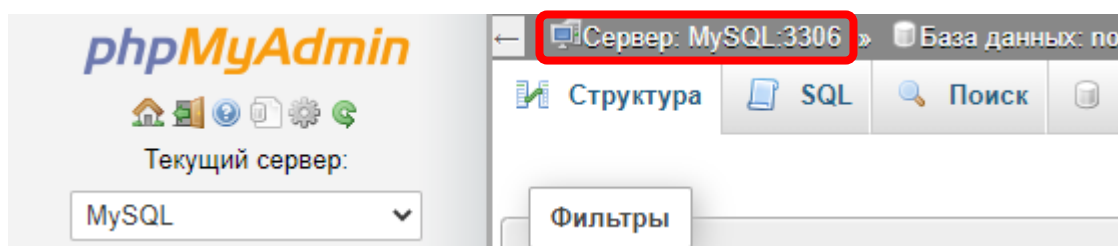


Рисунок 219 – Переход к настройкам сервера

Далее требуется выбрать вкладку «**Учетные записи пользователей**» (рис. 220) и кликнуть по названию учетной записи (в нашем случае «**root**»).

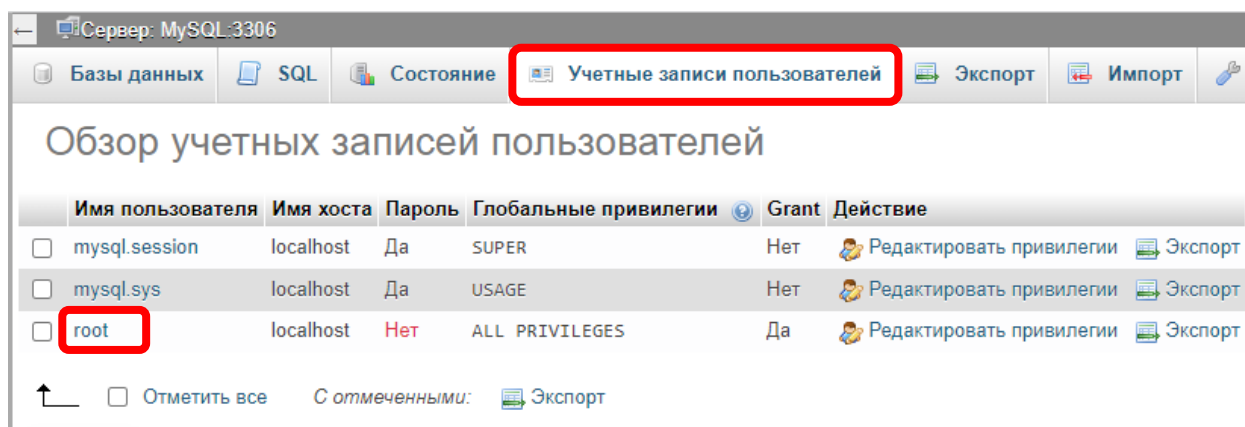


Рисунок 220 – Редактирование учетных записей пользователей

После этого необходимо перейти на вкладку «**Change password**» (рис. 221), где ввести новый пароль, а еще лучше нажать кнопку «Генерировать».

The screenshot shows the MySQL User Management web interface. At the top, there are tabs: 'Базы данных', 'SQL', 'Состояние', and 'Учетные записи пользователей'. Below these, there are buttons: 'Глобальные', 'База данных', 'Change password' (highlighted with a red rectangle), and 'Информация учётной записи'. The main heading is 'Редактирование привилегий: Учетная запись'. Below this is a warning message: 'Примечание: Вы пытаетесь изменить привилегии пользователя, под именем которого вы...'. The 'Изменить пароль' section has two radio buttons: 'Без пароля' and 'Пароль:' (selected). There are input fields for 'Введите:' and 'Подтверждение:', both masked with dots. A 'Стойкость:' indicator shows a green bar and the word 'Сложный'. Below this is a dropdown for 'Хеширование пароля:' with the selected option 'Собственная проверка подлинности MySQL'. At the bottom of this section, there is a 'Создать пароль' area with a 'Генерировать' button (highlighted with a red rectangle) and a text box containing the generated password '-l8U83Q3xKQIFKUr'. A 'Вперед' button is at the very bottom.

Рисунок 221 – Изменение пароля

2. Разграничение прав пользователей

При использовании **PHP** все указанные в его коде пароли доступа к базе данных скрываются от конечного пользователя, запрашивающего страницу через браузер. Тем не менее, пароли все еще остаются доступны всем программистам, работающим над проектом, а также администраторам **HTTP**-сервера. Серьезными неприятностями будут удаление всех данных из базы или копирование базы данных конкурентами. Для предотвращения подобных ситуаций необходимо ограничить права учетных записей, пароли от которых используются в коде. Не рекомендуется использовать прямое подключение на стороне пользователя (без **PHP**-прослойки), например, через **JavaScript**, т.к. в этом случае пароль становится доступен всем пользователям. В данном случае

необходимо разрешить доступ такой учетной записи только для выполнения запроса «**SELECT**».

В требуемой базе данных перейдем на вкладку «**Привилегии**» (рис. 222) и добавим новую учетную запись пользователя.

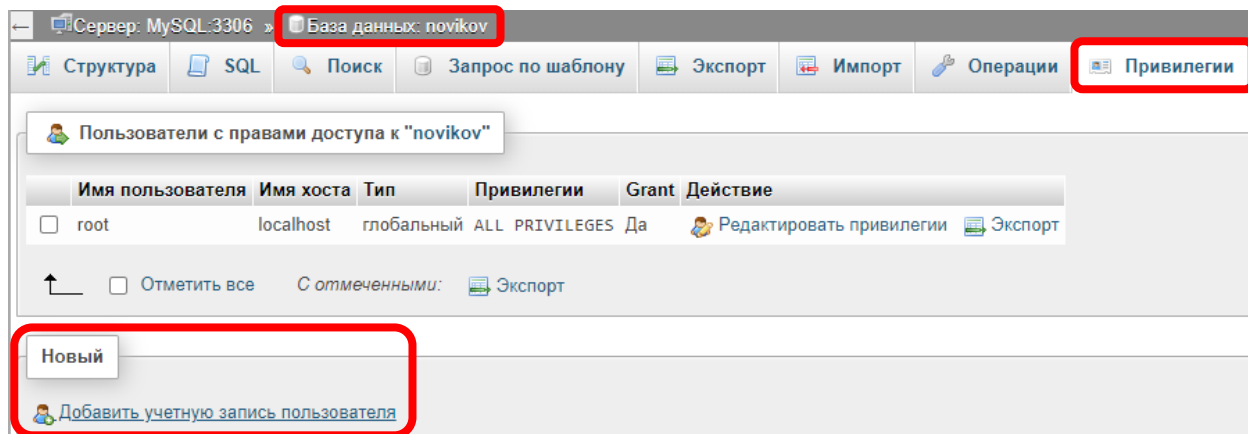


Рисунок 222 – Добавление нового пользователя

Нам необходимо создать две учетные записи «**view**» и «**edit**» для локального хоста (рис. 223).

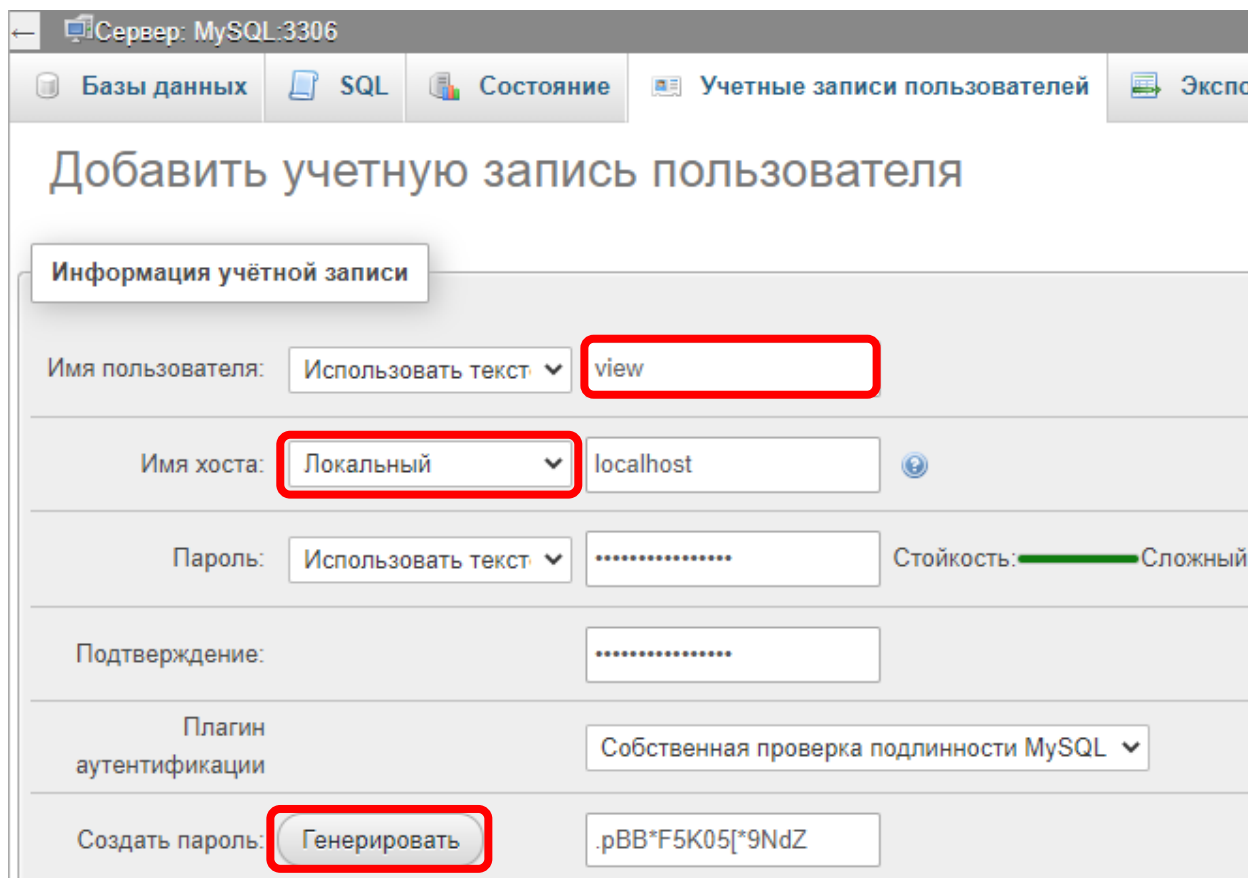


Рисунок 223 – Имя пользователя и пароль

Ниже необходимо указать права доступа. Для «**view**» это будет только «**SELECT**» (рис. 224), а для «**edit**» это будет «**SELECT**», «**INSERT**», «**UPDATE**» и, возможно, «**DELETE**».

Рисунок 224 – Выбор прав доступа

В итоге у нас получится **3** учетные записи (рис. 225), причем создавать и удалять таблицы можно будет только из учетной записи «**root**», а редактировать содержимое таблиц можно будет также и пользователю «**edit**».

Пользователи с правами доступа к "novikov"					
	Имя пользователя	Имя хоста	Тип	Привилегии	Grant
☐	edit	localhost	глобальный	SELECT, INSERT, UPDATE	Нет
☐	root	localhost	глобальный	ALL PRIVILEGES	Да
☐	view	localhost	глобальный	SELECT	Нет

Рисунок 225 – Добавленные учетные записи пользователей

7.2. *Оптимизация PHP-кода для MySQL

После того как мы сменили Логин и/или Пароль доступа к базе данных, необходимо изменить их и во всех **PHP**-файлах в строке:

```
//Подключиться к базе данных
$connect = mysqli_connect("localhost", "root", "", "Novikov");
```

Если таких **PHP**-файлов много, то придется вручную в каждом из них менять Логин и Пароль, что может быть затруднительно, а также этот ручной труд может приводить к ошибкам.

Переделаем код **PHP** таким образом, чтобы Логин и Пароль можно было вводить один раз для всех **PHP**-файлов. В папке «C:\wamp64\www\Novikov\Admin» создадим текстовый файл «**SQL_Connect.php**». В данный файл скопируем (из ранее созданных **PHP**-файлов) следующий код:

```
<?php
//Подключиться к базе данных
$connect = mysqli_connect("localhost", "root", "", "Novikov");
//Завершить работу в случае ошибки
if ($connect == false) {
    print("Невозможно подключиться к MySQL");
    exit;
}
//Установить кодировку
mysqli_set_charset($connect, 'utf8');
?>
```

В этом коде также нужно указать новые Логин и Пароль.

Далее во всех **PHP**-файлах, в которых содержался данный код подключения к **SQL**, необходимо заменить этот код единственной командой:

```
//Подключиться к базе данных
include "Admin/SQL_Connect.php";
```

Теперь, когда необходимо будет изменить Логин и Пароль для подключения к базе данных, достаточно будет сменить их в единственном файле «**SQL_Connect.php**».

7.3. *Использование MariaDB вместо MySQL

Ранее мы использовали **MySQL** в качестве базы данных по умолчанию. Но в этой роли также может выступать и **MariaDB**.

В изначальной конфигурации, **MySQL** подключается к порту **3306**, а **MariaDB** к порту **3307**. Проверить номера портов можно на главной странице **WAMP**-сервера (рис. 226), зайдя по адресу «localhost».

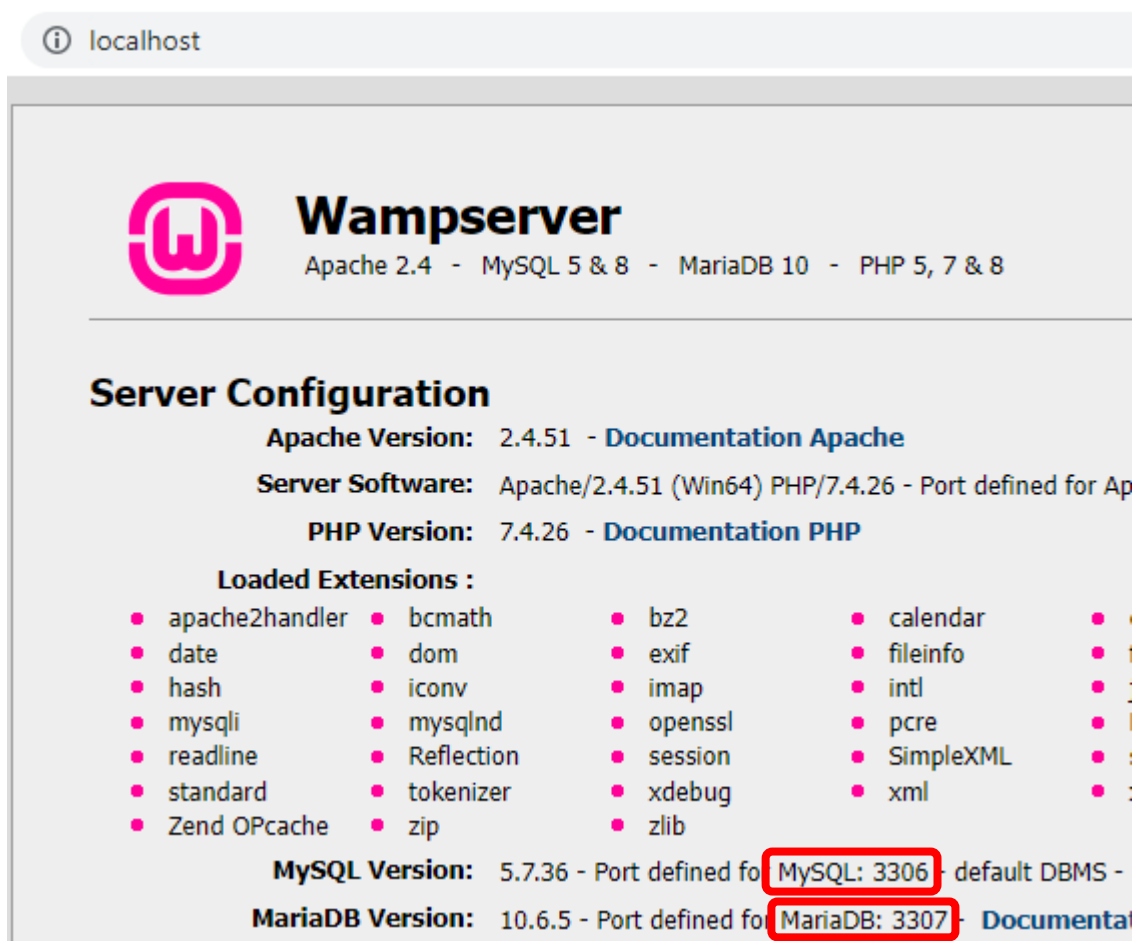


Рисунок 226 – Информация о портах MySQL и MariaDB

Для изменения конфигурации необходимо нажать **правую** клавишу мышки на значке **WAMP**-сервера (у часов) и в меню «Tools» выбрать пункт «**Invert default DBMS**» (рис. 227).

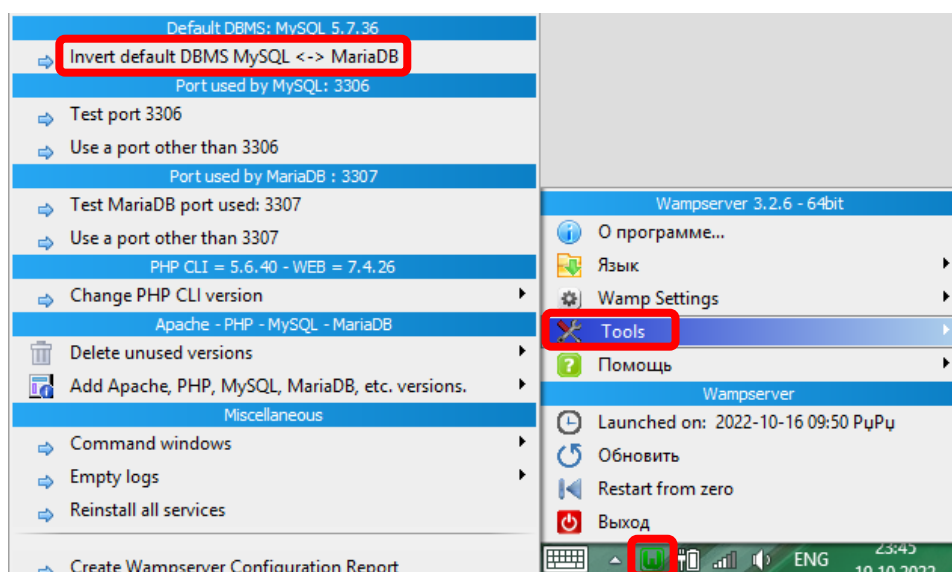


Рисунок 227 – Изменение БД по умолчанию

После этого порты изменятся (рис. 228): на **3306** – для **MariaDB** и **3308** – для **MySQL**.

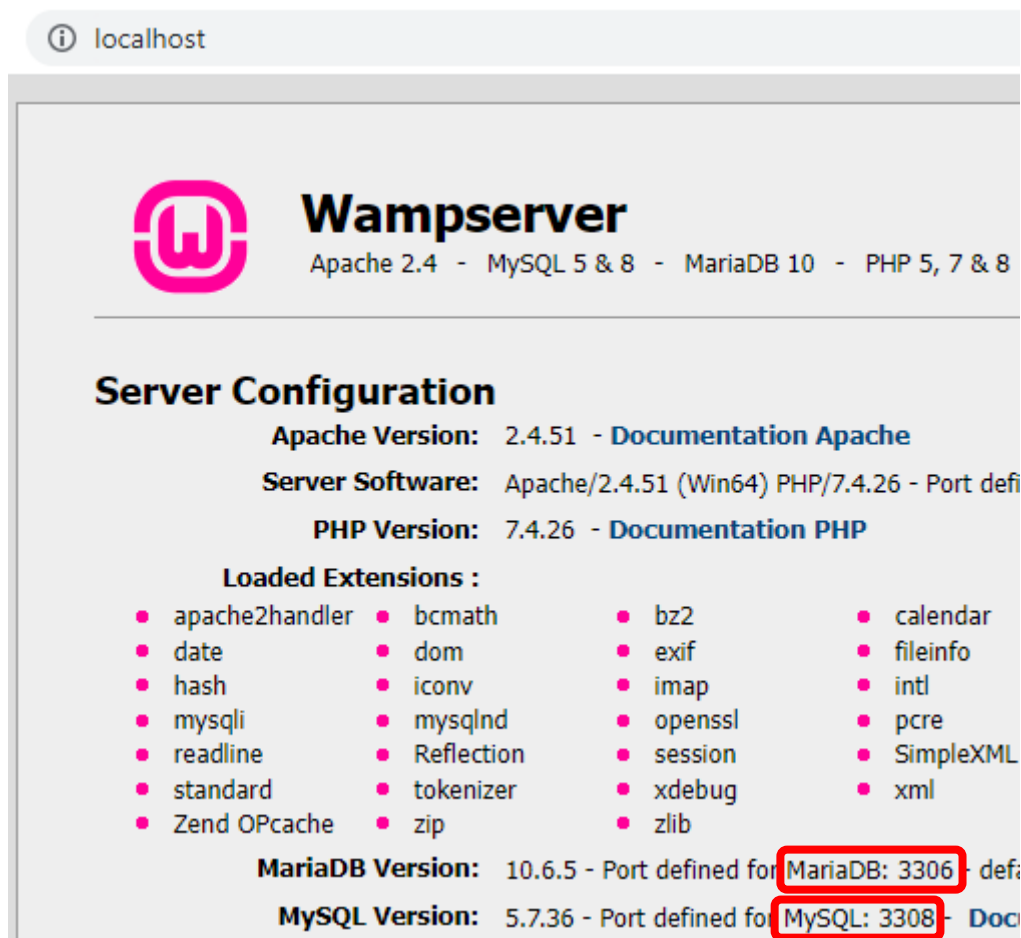


Рисунок 228 – Измененные порты MySQL и MariaDB

Кроме того, можно было вообще не «инвертировать» СУБД, а просто подключаться к необходимой по известному порту. Ранее для подключения к БД из **PHP** мы использовали команду:

```
//Подключение к СУБД по умолчанию  
mysqli_connect("localhost", "root", "", "Novikov");
```

У этой же команды есть и расширенный вариант с указанием номера порта:

```
//Подключение к СУБД по умолчанию  
mysqli_connect("localhost", "root", "", "Novikov", "3306");  
//Подключение к MariaDB  
mysqli_connect("localhost", "root", "", "Novikov", "3307");  
//Подключение к MySQL  
mysqli_connect("localhost", "root", "", "Novikov", "3308");
```

!!! Стоит обратить внимание, что одна и та же **СУБД не может занимать одновременно два порта**, т.е. если она выбрана как «СУБД по умолчанию» (на порту **3306**), то на портах **3307** или **3308** она будет недоступна!

7.4. *Файл настроек .htaccess

.htaccess (от англ. «hypertext access») – файл дополнительной конфигурации веб-сервера Apache (*а также подобных ему серверов*).

Файл **.htaccess** может быть размещен в любом каталоге сервера. Директивы этого файла действуют на все файлы в текущем каталоге и во всех его подкаталогах (*если эти директивы не переопределены директивами других файлов .htaccess, лежащих в этих подкаталогах*). Все изменения этого файла начинают действовать **мгновенно** после его сохранения (при следующем же обращении к странице). Позволяет задавать большое количество дополнительных параметров и разрешений для работы веб-сервера в отдельных директориях (папках, каталогах), таких как:

- закрыть директорию паролем;
- запретить доступ к файлам или папкам;
- исправить кодировку;
- назначить индексные страницы (страницы по умолчанию);
- запретить/разрешить выдачу листинга (списка файлов в каталогах);
- настроить редирект адресов (перенадресацию страниц, перенаправление страниц), в том числе сокращать адреса до более коротких и «красивых» или выдавать разные страницы пользователям с разными браузерами;
- установить страницы для отображения ошибок (403, 404 и т.п.);
- повысить эффективность сайта, управляя кешированием;
- и т.д...

К недостаткам применения файлов **.htaccess** стоит отнести снижение производительности сервера, т.к. при каждом обращении к странице ему нужно читать также и файл с настройками (*возможно даже сразу несколько файлов, если в вышестоящих папках также имеются файлы .htaccess*).

!!! Имя файла начинается с **точки** – так в **Linux** (и др. Unix-подобных системах) обозначаются **скрытые** файлы. Обычно они имеют специальное назначение. *Например, в ОС Android часто применяются файлы с именем «.nomedia». Они говорят системе о том, что изображения в текущей папке не нужно отображать в Галерее.*

!!! Файлы, начинающиеся с точки (в т.ч. **.htaccess**), не отдаются Apache-сервером клиенту, т.е. не будут доступны из браузера. *Также эти файлы не будут отображены в листинге (подробнее об этом см. раздел 7.5 «*Отображение содержимого папки»).*

Создание файла .htaccess в Windows

.htaccess является обычным текстовым файлом, который можно редактировать при помощи Блокнота.

Откроем новый документ в Блокноте, напомним в нем строчку «Options +Indexes» и сохраним файл с именем «.htaccess» в папку «C:\wamp64\www\Novikov».

!!! При этом стоит обратить внимание, что некоторые программы (включая стандартный Блокнот Windows) могут давать **неверное** имя итогового файла. Так, введя имя «.htaccess», в итоге мы получим файл с именем «.htaccess.txt», т.е. автоматически было добавлено расширение по умолчанию. Для решения этой проблемы необходимо при сохранении выбрать вариант «Все файлы (*.*)» (рис. 229).

!!! Кроме того, ОС Windows рассматривает часть имени после точки как расширение файла, поэтому некоторые версии Windows (такие как Windows 8 и более ранние) не могут правильно переименовать подобный файл, т.к. для них запись «.htaccess» означает расширение «htaccess», а имя файла в этом случае **пустое**. Но при этом они прекрасно далее работают с этими файлами, начинающимися с точки. Их можно копировать, перемещать и редактировать.

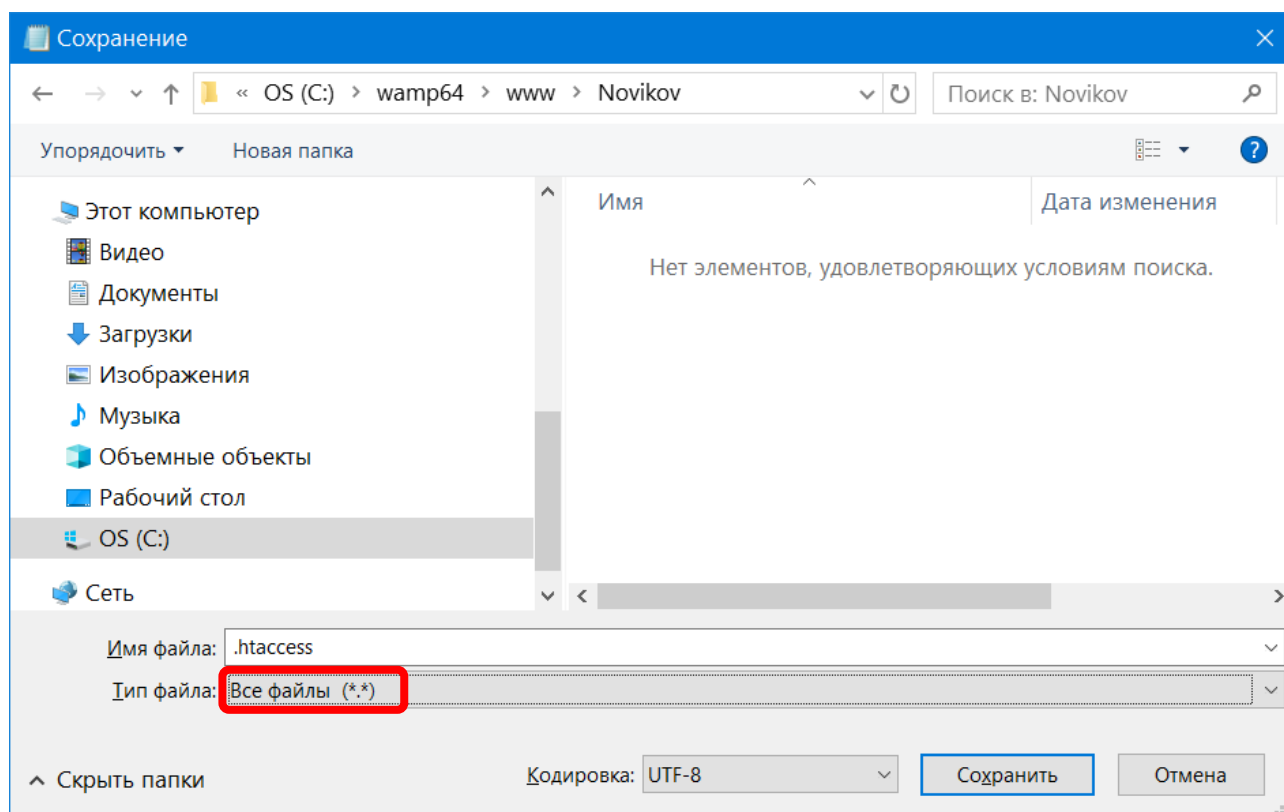


Рисунок 229 – Выбор типа файла

Комментарии в файлах .htaccess

В файле **.htaccess** можно писать однострочные комментарии (рис. 230), начинающиеся с символа #. Концом такого комментария будет конец текстовой строки. Эти же символы # можно применять, чтобы закомментировать часть кода, т.е. временно отключить часть директив.

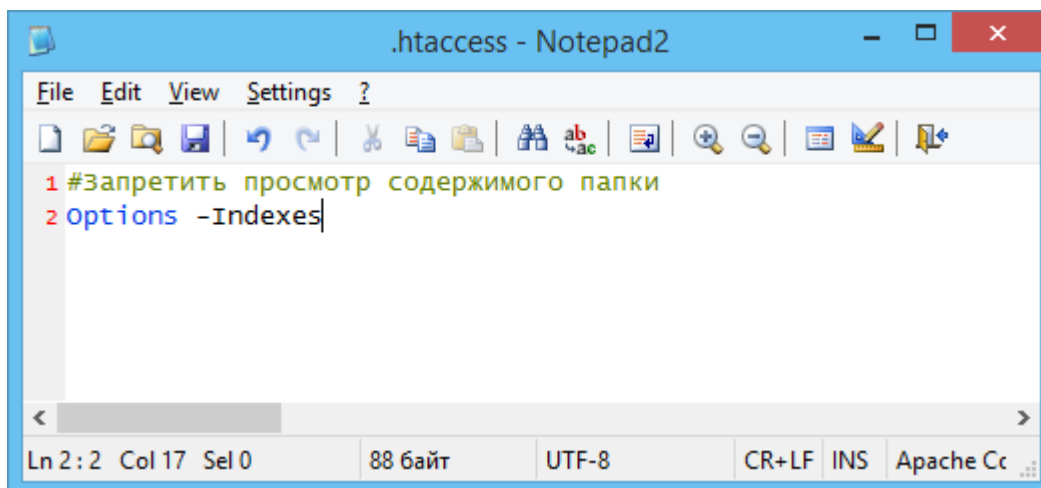


Рисунок 230 – Пример файла с комментариями

7.5. *Отображение содержимого папки (листинг и индекс)

Листинг файлов

Для отображения листинга файлов, т.е. для вывода списка файлов в текущей папке, необходимо прописать в **.htaccess** следующую строку:

#Просмотр содержимого папки
Options +Indexes

В большинстве случаев содержимое папок наоборот скрывается от пользователя в целях безопасности:

#Запретить просмотр содержимого папки
Options -Indexes

По умолчанию (если ничего не указано) один из вариантов «+» или «-» всегда наследуется от родительской папки или от глобальных настроек сервера.

Отображение для одной и той же папки при разных вариантах настройки продемонстрировано на рисунке 231.

Options -Indexes



Options +Indexes



Рисунок 231 – Варианты выключения/включения листинга

В листинге не будут отображены файлы, начинающиеся с точки (в т.ч. файл **.htaccess**), а также другие файлы, доступ к которым запрещен (см. раздел 7.11 «*Настройка прав доступа к Apache-серверу»).

Кроме того, можно скрыть файлы (и папки) из листинга, директивой:

#Скрыть файлы и папки из листинга

`IndexIgnore *.html DateTime.php Img`

Но такие файлы все еще будут доступны по их прямому адресу.

Индексный файл

Index.html («указатель».html) – традиционное название файла, используемого в качестве **стартовой страницы** для веб-сайта или папки на сайте. Кроме **index.html**, в качестве имени индекса используется **index.php**, а также файлы **index.*** с некоторыми другими расширениями. *Иногда в роли индексных файлов также выступают **default.html** или т.п.*

Если, например, в папке «<http://localhost/Novikov/>» существует файл **index.html**, то можно открывать его как по полному адресу «<http://localhost/Novikov/index.html>», так и по короткому адресу папки «<http://localhost/Novikov/>». *Если для папки включен листинг (см. рис. 231), то в этом случае он не будет отображаться.*

Таким образом, чтобы какая-то страница стала «главной» на сайте, достаточно переименовать ее в «**index.php**».

Изменение индексного файла

Также можно легко изменить имя индексного файла на любое другое. Для этого в **.htaccess** достаточно написать:

#Изменить индексный файл

`DirectoryIndex Lab4.php`

Теперь данная страница будет доступна как по адресу «<http://localhost/Novikov/Lab4.php>», так и по короткому адресу «**<http://localhost/Novikov/>**». *Естественно, что файл «<http://localhost/Novikov/Lab4.php>» должен физически существовать.*

Данная директива отменит все предыдущие настройки, т.е. файл «**index.html**» (или подобные) больше не будет являться индексным файлом. Но можно указать и сразу несколько индексных файлов, например:

`DirectoryIndex Lab4.php index.php index.html`

В этом случае первый из указанных файлов будет выступать в качестве индексного, но если этот файл не существует, то индексом станет второй, третий и т.д.

!!! Если ни одного из индексных файлов не существует, то в качестве индекса отобразится листинг файлов (см. рис. 231), если конечно он не отключен.

7.6. *Настройка кэширования файлов

Кэширование удобно тем, что снижает нагрузку на сервер. Без кэша пользователь загружал бы с сервера все требуемые файлы при каждом обращении к странице, включая ее обновления (F5).

Кэширование происходит не на стороне сервера (хотя и настраивается на сервере), а в браузерах пользователей – сервер только дает на это команду (разрешение). Недостатком кэширования является то, что при изменении файлов на сервере, пользователи увидят это не сразу (даже если нажмут F5 в браузере). *Особенно это неудобно для самого разработчика, т.к. после внесения изменений он хочет их сразу протестировать. В этом случае необходимо нажимать Ctrl+F5 вместо F5 в браузере – т.е. обновлять страницы и все используемые файлы с сервера, а не из кэша браузера.*

Для включения кэширования необходимо прописать в файле **.htaccess** следующие команды:

```
#Настройка кэширования
<FilesMatch "\.(ico|jpe?g|png|gif)$">
    #Кэшируем изображения на 30 дней
    Header set Cache-Control "max-age=2592000"
</FilesMatch>
<FilesMatch "\.(css)$">
    #Кэшируем стили на 1 день
    Header set Cache-Control "max-age=86400"
</FilesMatch>
<FilesMatch "\.(js)$">
    #Кэшируем скрипты на 2 часа
    Header set Cache-Control "max-age=7200"
</FilesMatch>
<FilesMatch "\.(html?)$">
    #Кэшируем статические страницы на 1 минуту
    Header set Cache-Control "max-age=60"
</FilesMatch>
```

Типы файлов указываются в FilesMatch (перечисляются через символ «|»). Время кеширования задается в секундах (для директивы «max-age» HTTP-заголовка «**Cache-Control**»).

О том, как увидеть HTTP-заголовки рассказывается в разделе 7.10 «**Просмотр кодов ответов и HTTP-заголовков в браузере».

Возможные ошибки

После описанной выше настройки при обращении к странице может появиться внутренняя ошибка сервера (рис. 232). Такое возникает, когда не все требуемые для работы модули включены.

Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator at wampserver@wampserver.invalid to inform them of the time this error occurred, and the actions you performed just before this error.

More information about this error may be available in the server error log.

Apache/2.4.51 (Win64) PHP/7.4.26 Server at localhost Port 80

Рисунок 232 – Внутренняя ошибка сервера (500)

В данном случае нам необходимо включить модуль для работы с заголовками «**headers_module**» (рис. 233). Для этого необходимо нажать **левую** клавишу мышки на значке WAMP-сервера в трее, после чего выбрать «Apache» → «Apache modules» («Модули Apache»). После этого обновляем страницу в браузере.

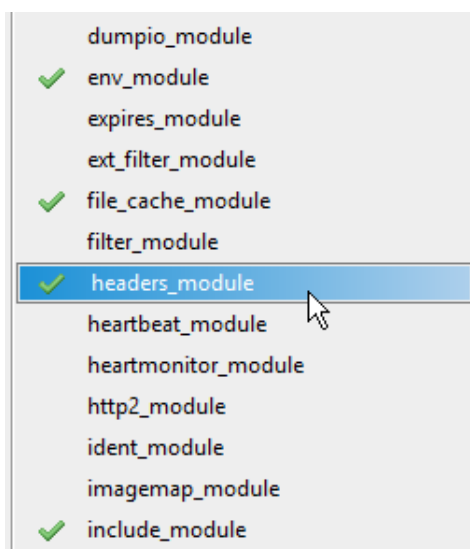


Рисунок 233 – Включение модуля заголовков

7.7. *Простая переадресация

Для переадресации страницы необходимо прописать в файле **.htaccess** следующую строку:

#Переадресация

Redirect 301 "/Работа_3" "/Lab3.php"

Теперь одна и та же страница станет доступна как по адресу «<http://localhost/novikov/Lab3.php>», так и по адресу «http://localhost/novikov/Работа_3». При этом реально существует только страница **Lab3.php**.

Подробнее про переадресацию и Коды 301, 302 и т.д., см. раздел **7.9** «**Коды ошибок HTTP (404 и др.)».

7.8. *Отображение страниц ошибок

Араше-сервер отображает стандартное сообщение об ошибке 404 «Не найдено», представленное на рисунке 234. Аналогичные сообщения, но со своим текстом, отображаются и для других ошибок.

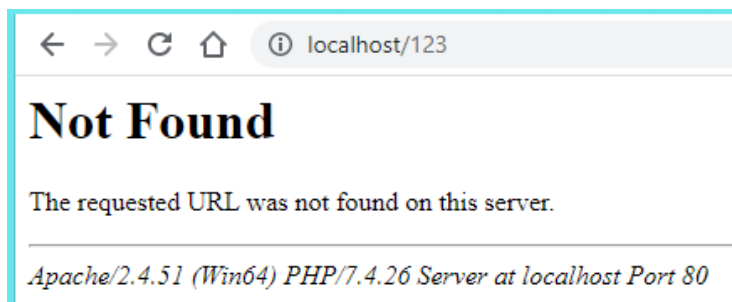


Рисунок 234 – Стандартный ответ 404 Араше-сервера

Обычно владельцы сайта хотят, чтобы страница была менее отпугивающей и более красочной, содержала логотип, название сайта и кнопку перехода на главную страницу (рис. 235). *Сайты магазинов заточены под продажи, поэтому даже на страницу ошибки они стремятся добавить предложения товаров и/или строку поиска товаров.*

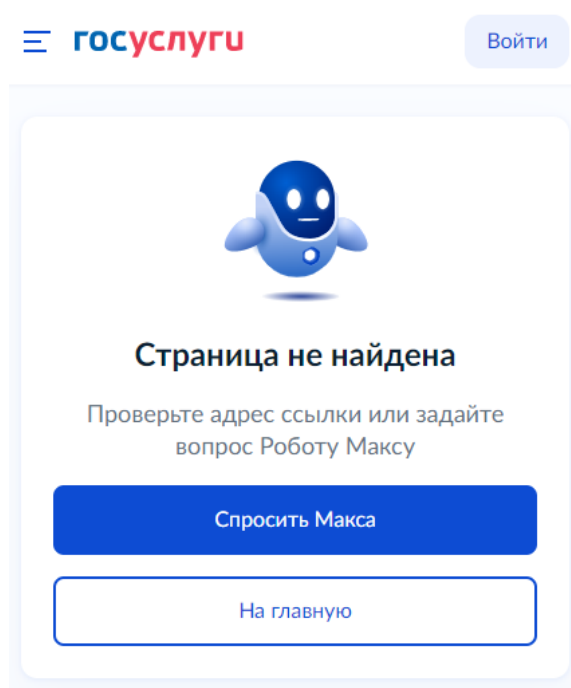


Рисунок 235 – Ошибка на сайте Госуслуг

Для реализации этой функции, необходимо создать HTML-страницу для отображения ошибки (в нашем примере это будет «error404.html», расположенный в папке «C:\wamp64\www»), после чего в файле **.htaccess** прописать следующую строку:

#Сообщения об ошибках

ErrorDocument 404 /error404.html

Аналогично можно создать и настроить страницы и для других ошибок

ErrorDocument 403 /error403.html

...

Подробнее про Коды ошибок см. раздел **7.9 «**Коды ошибок HTTP (404 и др.)»**.

Если мы не создаем отдельную страницу, а при ошибке хотим отображать главную страницу, то нужно написать:

ErrorDocument 404 /

7.9. **Коды ошибок HTTP (404 и др.)

Пожалуй, самой известной из ошибок (рис. 236, 237) является «**404 Not Found**» («Не найдено»).

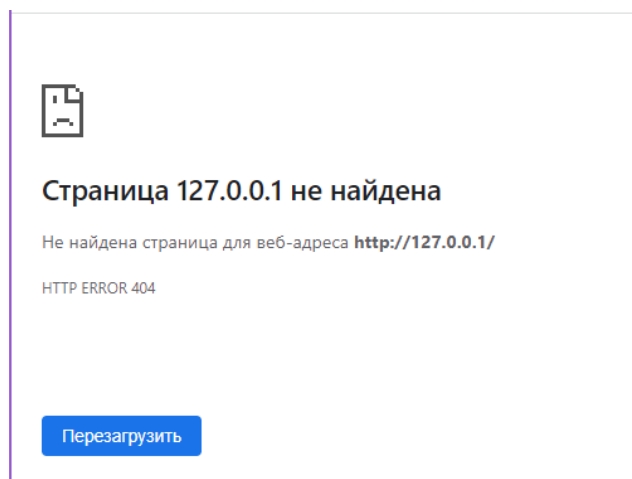


Рисунок 236 – Ошибка 404

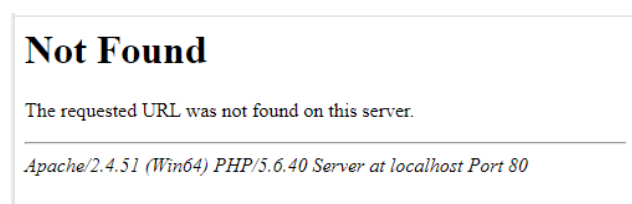


Рисунок 237 – Ошибка 404 Apache-сервера

На самом деле, правильное название для этого значения не «ошибка», а «**Код состояния**», и гораздо более частым ответом сервера является «**200 ОК**» («Хорошо»), при котором мы получаем запрошенную страницу. Просто Коды состояний **4xx** и **5xx** имеют собственные страницы для отображения, поэтому мы их и знаем, а когда все в порядке (**1xx**, **2xx**, **3xx**), то мы попадаем на требуемую страницу, не видя полученного кода.

Код состояния HTTP (HTTP status code)[5], **Код ответа HTTP** – часть первой строки ответа сервера при запросах по протоколу HTTP. Представляет собой целое десятичное число из трех цифр. Первая цифра указывает на **класс**

кода состояния. За кодом ответа обычно следует отделенная пробелом поясняющая фраза на английском языке, которая разъясняет человеку причину именно такого ответа.

Существует 5 **классов** кодов состояний:

- **1xx**: Informational («Информационные»);
- **2xx**: Success («Успешно»);
- **3xx**: Redirection («Перенаправление»);
- **4xx**: Client Error («Ошибка клиента»);
- **5xx**: Server Error («Ошибка сервера»).

Только для двух последних из них (**4xx** и **5xx**) в браузере отображается сообщение об ошибке.

Клиент узнает по коду ответа о результатах своего запроса и определяет, какие действия ему предпринимать дальше. Клиент может и не знать все коды состояния (например, клиент использует старую версию ПО, или сервер применяет нестандартные коды), но он обязан отреагировать в соответствии с **классом** кода.

Коды состояний описаны в стандартах **RFC** (Request for Comments – пронумерованные документы, содержащие технические спецификации и стандарты Интернета) утвержденных **IETF** (Инженерным советом Интернета).

Список кодов состояния

Далее рассмотрим названия некоторых из кодов состояния, самые «интересные» из которых выделены жирным. Некоторые названия будут снабжены более подробной информацией[5].

«Информационные» (**1xx**) и «Успешные» (**2xx**) коды приведены в таблице 1.

Таблица 1 – Коды 1xx и 2xx

1xx: Informational («Информационные»):	
100 Continue	«Продолжайте»
101 Switching Protocols	«Переключение протоколов»
102 Processing	«Идёт обработка»
103 Early Hints	«Ранняя метайнформация»
2xx: Success («Успешно»):	
200 OK	«Хорошо» Успешный запрос. Если клиентом были запрошены какие-либо данные, то они находятся в заголовке и/или теле ответа.
201 Created	«Создано»
202 Accepted	«Принято»
203 Non-Authoritative Information	«Информация не авторитетна» Аналогично ответу 200, но отдаваемая информация взята не из первичного источника (а, например, из резервной копии, другого сервера или т.п.) и поэтому может быть неактуальна.
204 No Content	«Нет содержимого»
205 Reset Content	«Сбросить содержимое»
206 Partial Content	«Частичное содержимое»

Коды «Перенаправления» (3xx) представлены в таблице 2.

Таблица 2 – Коды 3xx

3xx: Redirection «Перенаправление»:	
300 Multiple Choices	«Множество выборов» По указанному адресу существует несколько вариантов предоставления ресурса (по типу MIME, по языку или по другим характеристикам), из которых клиент может выбирать наиболее подходящий.
301 Moved Permanently	«Перемещено навсегда» Возникает в ситуации, когда запрошенный ресурс был на постоянной основе перемещен в новое место, и указывает на то, что клиент должен обновить у себя текущую ссылку. Адрес нового месторасположения ресурса указывается в HTTP-заголовке Location получаемого ответа. <i>С точки зрения SEO, именно код состояния 301 сообщает поисковым роботам, что нужно объединить два разных адреса в один (где основным будет тот, на который происходит перенаправление). Поисковые системы также рекомендуют настраивать данное перенаправление с дополнительных зеркал на основное, например, когда сайт одновременно доступен по адресам с www и без www или использует защищенный протокол https, но также доступен по http.</i>
302 Found, 302 Moved Temporarily	«Найдено», «Временно перемещено» Запрошенный документ временно доступен по другому адресу, указанному в HTTP-заголовке Location получаемого ответа. <i>В отличие от 301, в данном случае клиенту не требуется обновлять у себя ссылку, т.к. ресурс скоро вернется по прежнему адресу.</i>
303 See Other	«Смотреть другое»
304 Not Modified	«Не изменялось» Указывает, что ресурс не был изменен (с версии, указанной заголовками запроса If-Modified-Since или If-None-Match). В таком случае нет необходимости повторно передавать ресурс, поскольку у клиента все еще есть ранее загруженная копия. <i>Сервер возвращает такой код, если клиент запросил документ используя заголовки If-Modified-Since или If-None-Match и документ не изменился с указанного момента. При этом сообщение сервера не содержит тела.</i>
305 Use Proxy	«Использовать прокси»
307 Temporary Redirect	«Временное перенаправление» Запрашиваемый ресурс на короткое время доступен по другому адресу, указанному в поле Location заголовка. <i>Практически аналогичен 302.</i>
308 Permanent Redirect	«Постоянное перенаправление» Запрашиваемый ресурс был окончательно перенесен на новый адрес, указанный в поле Location заголовка. <i>Практически аналогичен 301.</i>

Коды ошибок (**4xx**, **5xx**) представлены в таблице 3.

Таблица 3 – Коды 4xx и 5xx

4xx: Client Error («Ошибка клиента»):	
400 Bad Request	«Неправильный, некорректный запрос» Сервер обнаружил в запросе клиента синтаксическую ошибку.
401 Unauthorized	«Не авторизован» Аналогично «403 Forbidden», но используется, когда требуется аутентификация, которая не пройдена или еще не была предоставлена. <i>Для доступа к запрашиваемому ресурсу требуется аутентификация (клиент должен представиться). Если запрос уже включает данные для авторизации, ответ 401 означает, что в авторизации с ними отказано.</i>
402 Payment Required	«Необходима оплата» – зарезервировано для использования в будущем.
403 Forbidden	«Запрещено» («Не уполномочен») Сервер понял запрос, но он отказывается его выполнять из-за ограничений в доступе для клиента (т.е. клиент не уполномочен совершать операции с запрошенным ресурсом). Если для доступа к ресурсу требуется аутентификация средствами HTTP, то сервер вернет ответ 401 (либо 407 при использовании прокси). В любом случае, серверу следует сообщить причины отказа в обработке запроса. Наиболее вероятными причинами ограничения может послужить попытка доступа к системным ресурсам веб-сервера (например, к файлам .htaccess или .htpasswd) или к файлам, доступ к которым был закрыт с помощью конфигурационных файлов, либо сервер не удовлетворен IP-адресом клиента (например, при блокировках). <i>Сервер возвращает HTTP 401, когда клиент не прошел аутентификацию, и подразумевает, что успешный ответ может быть возвращен после действительной аутентификации, а HTTP 403 означает, что клиенту не разрешен доступ к ресурсу, несмотря на предоставление аутентификации.</i>

404 Not Found	«Не найдено» Самая распространенная ошибка при использовании Интернета. Основные причины ошибки – неверное написание адреса страницы пользователем или переход по так называемым «битым» ссылкам. Сервер понял запрос, но не нашел соответствующего ресурса по указанному URL. Если серверу известно, что по этому адресу был документ, то ему желательно использовать код 410. Ответ 404 может использоваться вместо 403, если требуется тщательно скрыть от посторонних глаз определенные ресурсы. Некоторые сайты вместо 404 возвращают стандартную страницу ошибки, но с кодом «200 ОК» (иногда даже просто открывают главную страницу сайта), это известно как «мягкая» ошибка 404. Мягкий 404 является проблемой для поисковых систем и для программ, применяемых администраторами сайта, для определения «битых» ссылок. Также мягкий 404 может произойти в результате ошибки конфигурации, например, веб-сервере Apache неверно выдаст 200, когда путь к странице ошибки (указанный в .htaccess) задан как абсолютный (например, «http://example.com/error.html»), т.к. на итоговую страницу мы будем попадать через редирект 302. Но правильно выдаст 404 когда путь задан как относительный («/error.html»).
405 Method Not Allowed	«Метод не поддерживается»
406 Not Acceptable	«Неприемлемо»
408 Request Timeout	«Истекло время ожидания», «Тайм-аут запроса» Время ожидания сервером передачи от клиента истекло. Клиент может повторить аналогичный предыдущему запрос в любое время.
409 Conflict	«Конфликт»
410 Gone	«Удалён» Указывает, что запрошенный ресурс ранее существовал по этому адресу, но больше недоступен и не будет доступен снова. Серверу в этом случае неизвестно местоположение альтернативного документа (в отличие от 301 или 302). Используется, когда ресурс был намеренно удален. Получив код 410, клиент не должен запрашивать ресурс в будущем, а поисковые системы должны удалить ресурс из своих индексов. Обычно вместо этого используется «404 Not Found».

411 Length Required	«Необходима длина» В запросе не указана длина его содержимого, требуемая запрашиваемым ресурсом. <i>Для указанного ресурса клиент должен указать Content-Length в заголовке запроса. Без указания этого поля не стоит делать повторную попытку запроса к серверу по данному адресу. Такой ответ естественен для запросов типа POST и PUT, если по указанному адресу производится загрузка файлов, а на сервере стоит ограничение на их объем, тогда разумней будет проверить заголовок Content-Length в самом начале, и сразу отказать в загрузке, чем провоцировать бессмысленную нагрузку, разрывая соединение, когда клиент действительно пришлет слишком объемное сообщение.</i>
412 Precondition Failed	«Условие ложно»
413 Payload Too Large	«Полезная нагрузка слишком велика»
414 URI Too Long	«URI слишком длинный» Сервер не может обработать запрос из-за слишком длинного указанного адреса (URI). <i>Такую ошибку можно спровоцировать, например, когда клиент пытается передать длинные параметры через метод GET, а не POST.</i>
415 Unsupported Media Type	«Неподдерживаемый тип данных»
416 Range Not Satisfiable	«Диапазон не достижим»
417 Expectation Failed	«Ожидание не оправдалось»
418 I'm a teapot	«Я – чайник» Этот код был введен в 1998 году как одна из традиционных первоапрельских шуток авторов стандарта. Данный код поддерживается большинством реальных серверов, хотя его реализация не требуется. RFC указывает, что этот код должен возвращаться чайниками, которым было предложено сварить кофе. <i>Иногда этот код статуса также используется в качестве ответа на заблокированный запрос вместо более подходящего 403 Forbidden.</i>

451 Unavailable For Legal Reasons	«Недоступно по юридическим причинам» Оператор сервера получил законное требование отказать в доступе к ресурсу, например, от органов государственной власти, суда или по требованию правообладателя (в случае нарушения авторских прав). Можно сказать, что код HTTP 451 является уточнением для кода HTTP 403. Ранее (<i>а часто и до сих пор</i>) именно 403 («Запрещено») использовался для закрытия ресурса, при этом 403 не поясняет причину закрытия. В свою очередь, 451 дополнительно к коду ответа использует заголовок Link, уточняющий, кто именно блокирует информацию. <i>Код ошибки является отсылкой к антиутопическому роману Рэя Брэдбери «451 градус по Фаренгейту», в котором книги объявлены вне закона (451 градус – температура горения бумаги).</i> Также этот код используется, если контент не может быть показан в стране пользователя из-за договорных или лицензионных ограничений с владельцем контента, или когда издатель сам отказывается предоставлять контент пользователю, поскольку в стране пользователя действуют нормативные требования, которые издатель не собирается соблюдать, например, веб-сайты, расположенные за пределами ЕС, могут отказаться предоставлять контент пользователям в ЕС, поскольку они не хотят соблюдать GDPR (регламент защиты персональных данных – постановление Европейского союза, которое подразумевает штрафы в 20 млн. евро).
5xx: Server Error («Ошибка сервера»):	
500 Internal Server Error	«Внутренняя ошибка сервера» Общее сообщение об ошибке, выдаваемое при возникновении непредвиденной ситуации и отсутствии более конкретного сообщения. <i>Например выдается, когда есть ошибки в файле .htaccess.</i>
501 Not Implemented	«Не реализовано» Сервер либо не распознает метод запроса, либо не имеет возможности выполнить запрос. Обычно это подразумевает будущую доступность (например, новую функцию API веб-сервиса).
502 Bad Gateway	«Плохой шлюз», «Ошибочный шлюз» Сервер действовал как шлюз или прокси-сервер и получил недействительный ответ от вышестоящего сервера.

503 Service Unavailable	«Сервис недоступен» Сервер временно не имеет возможности обрабатывать запросы по техническим причинам (обслуживание, перегрузка и прочее). Обычно это временное состояние. В поле Retry-After HTTP-заголовка, сервер может указать время, через которое клиенту рекомендуется повторить запрос. <i>Хотя во время перегрузки очевидным кажется сразу разрывать соединение, эффективней может оказаться установка большого значения поля Retry-After для уменьшения частоты избыточных запросов.</i>
504 Gateway Timeout	«Шлюз не отвечает» Сервер действовал как шлюз или прокси-сервер и не получил своевременного ответа от вышестоящего сервера.
505 HTTP Version Not Supported	«Версия HTTP не поддерживается» Сервер не поддерживает версию HTTP, использованную в запросе.

7.10. **Просмотр кодов ответов и HTTP-заголовков в браузере

Для того чтобы увидеть код ответа сервера и HTTP-заголовки, необходимо проделать следующее:

- открыть в браузере требуемую страницу (подойдет большинство браузеров, таких как: Chrome, Firefox, Яндекс Браузер и многие др.);
- нажать клавишу «F12» (либо комбинацию клавиш «Ctrl+Shift+I»);
- перейти на вкладку «Network»/«Сеть» (рис. 238);

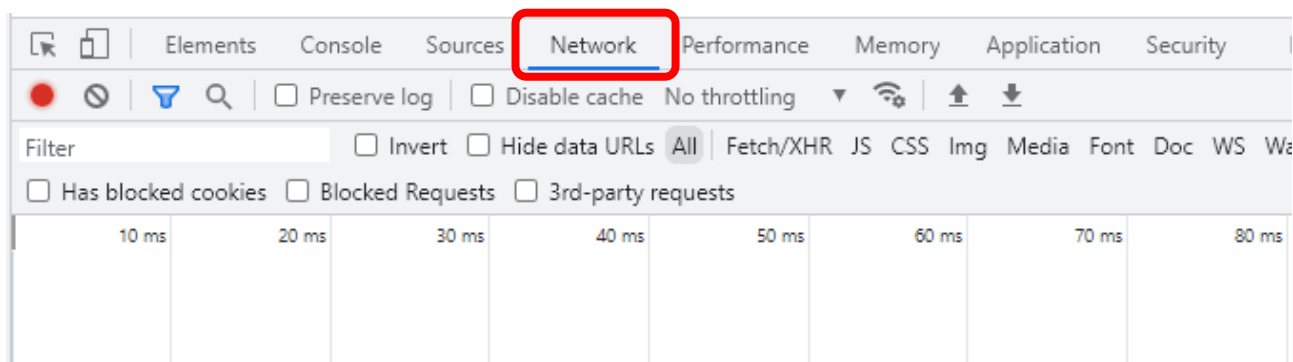


Рисунок 238 – Пример в браузере Chrome

- обязательно обновить страницу («F5»);
- !!! Лучше использовать «Ctrl+F5» – если мы не хотим, чтобы данные загружались из кэша браузера, а полностью обновились с сервера.

— выбрать требуемую страницу (или файл) и для него открыть вкладку «**Headers**»/«**Заголовки**» (рис. 239).

Теперь мы можем увидеть код и заголовки ответа сервера (рис. 239). Если пролистать ниже, то можно также увидеть и заголовки запроса.

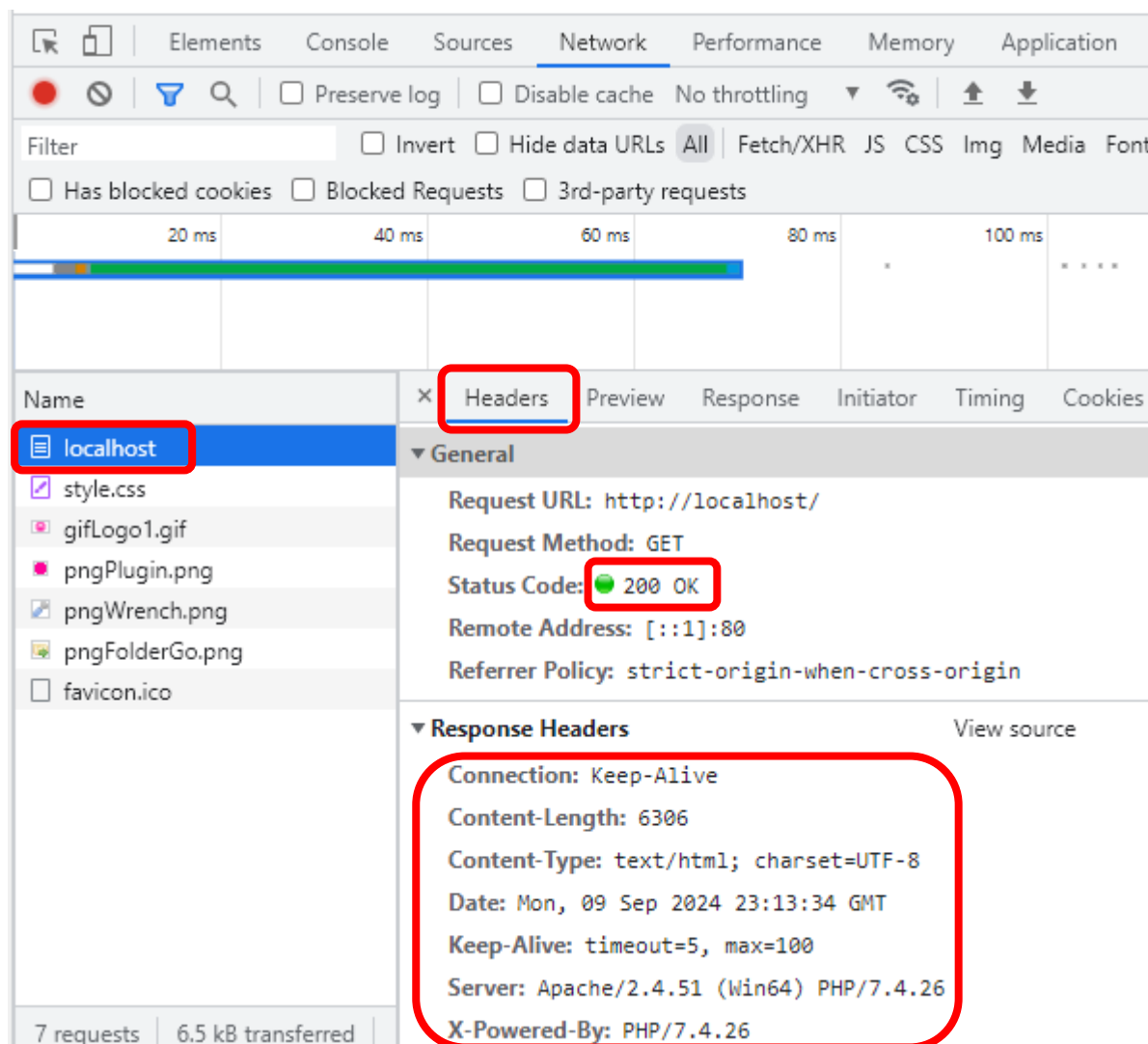


Рисунок 239 – Код ответа и заголовки HTTP

Можно просматривать заголовки и в исходном (необработанном) виде, для этого имеется соответствующий переключатель (рис. 240).

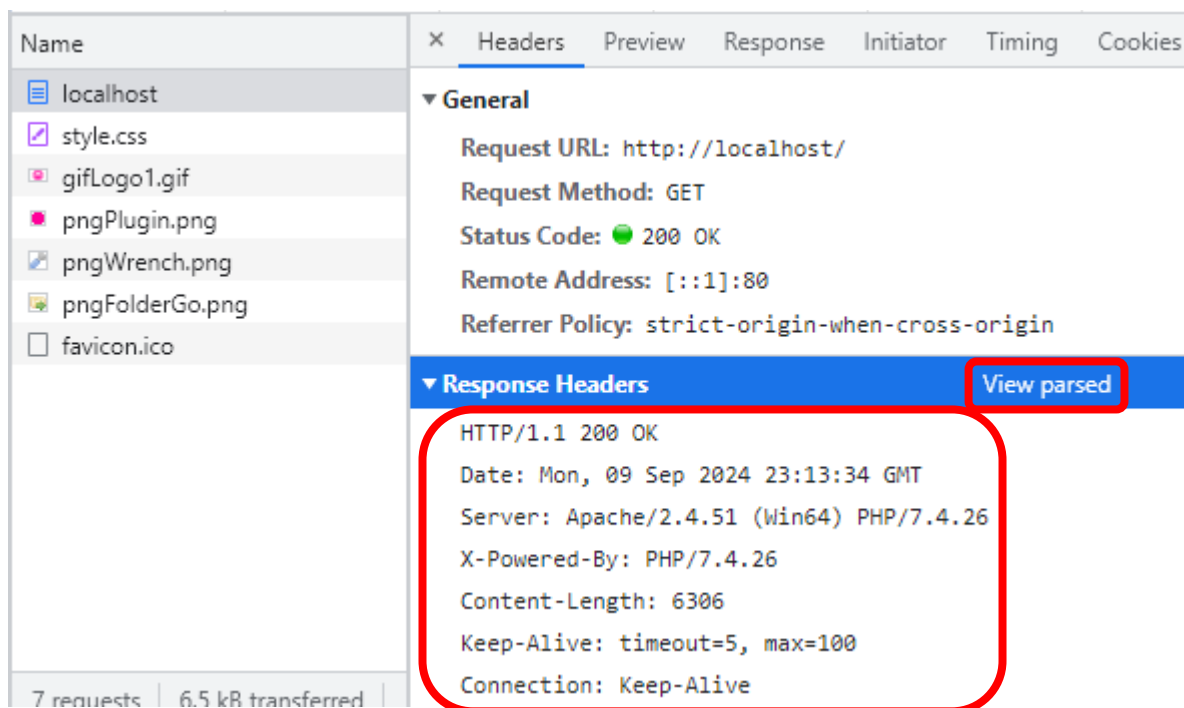


Рисунок 240 – Заголовки в необработанном виде

7.11. *Настройка прав доступа к Apache-серверу

1. Запрет доступа к файлам

Для полного запрета доступа к файлам или папкам с определенными именами достаточно прописать в файле **.htaccess** следующее:

#Запрет доступа к файлам в текущей папке

```
<Files ~ \"^(Секрет1.txt|Секрет2.html)$">
```

```
Deny from all
```

```
</Files>
```

Защищаемые файлы перечисляются через символ «|» (в данном примере это «Секрет1.txt» и «Секрет2.html»).

Запрет доступа для определенных типов файлов выполняется следующим образом:

#Запрет доступа к типам файлов

```
<FilesMatch \"\.(txt|jpe?g)$">
```

```
Deny from all
```

```
</FilesMatch>
```

!!! В обоих случаях, защищаемые файлы также скрываются и из листинга, если он имеется (*подробнее об этом см. раздел 7.5 «*Отображение содержимого папки»*).

2. Закрывание директории паролем

Для закрытия директории паролем необходимо создать файл **.htaccess** **внутри** защищаемой папки. В нашем случае имеет смысл закрыть паролем

папку «Admin» («C:\wamp64\www\Novikov\Admin», «http://localhost/Novikov/Admin/») и, соответственно, именно в ней создать данный файл. В этом файле **.htaccess** прописываем следующее:

#Закреть директорию паролем

AuthName "Authorization"

AuthType Basic

AuthUserFile C:\wamp64\www\Novikov\.htpasswd

require valid-user

где «C:\wamp64\www\Novikov\» – путь к файлу с паролями, который будет получен далее.

Теперь при входе по адресу «http://localhost/Novikov/Admin/» мы увидим окно с предложением ввести пароль от папки (рис. 241).

Рисунок 241 – Ввод пароля от папки

3. Файл паролей

.htpasswd (с точкой перед именем!) – файл, содержащий пароли для веб-сервера Apache (и некоторых других, например, nginx). Метод авторизации с использованием такого файла называется базовым (basic authentication).

Файл состоит из строк, каждая из которых соответствует паре логин/пароль (точнее хеш пароля, т.е. пароль «зашифрован» от посторонних глаз). Логин отделяется от пароля двоеточием:

Alex:\$apr1\$XNJ/LybF\$fX5QG2bogUahnO6DpPvif/

Выглядит страшно, но это всего лишь пароль «123456».

4. Создание файла паролей

Для создания файла паролей служит утилита **htpasswd** (без точки перед именем!), входящая в состав Apache-сервера.

Перейдем в папку «C:\wamp64\bin\apache\apache2.4.51\bin\», где прямо в адресной строке Проводника необходимо ввести (рис. 242):

htpasswd -c C:\wamp64\www\Novikov\.htpasswd Alex

где **-c** – команда на создание нового файла;
C:\.\.htpasswd – создаваемый файл;
Alex – имя добавляемого пользователя.

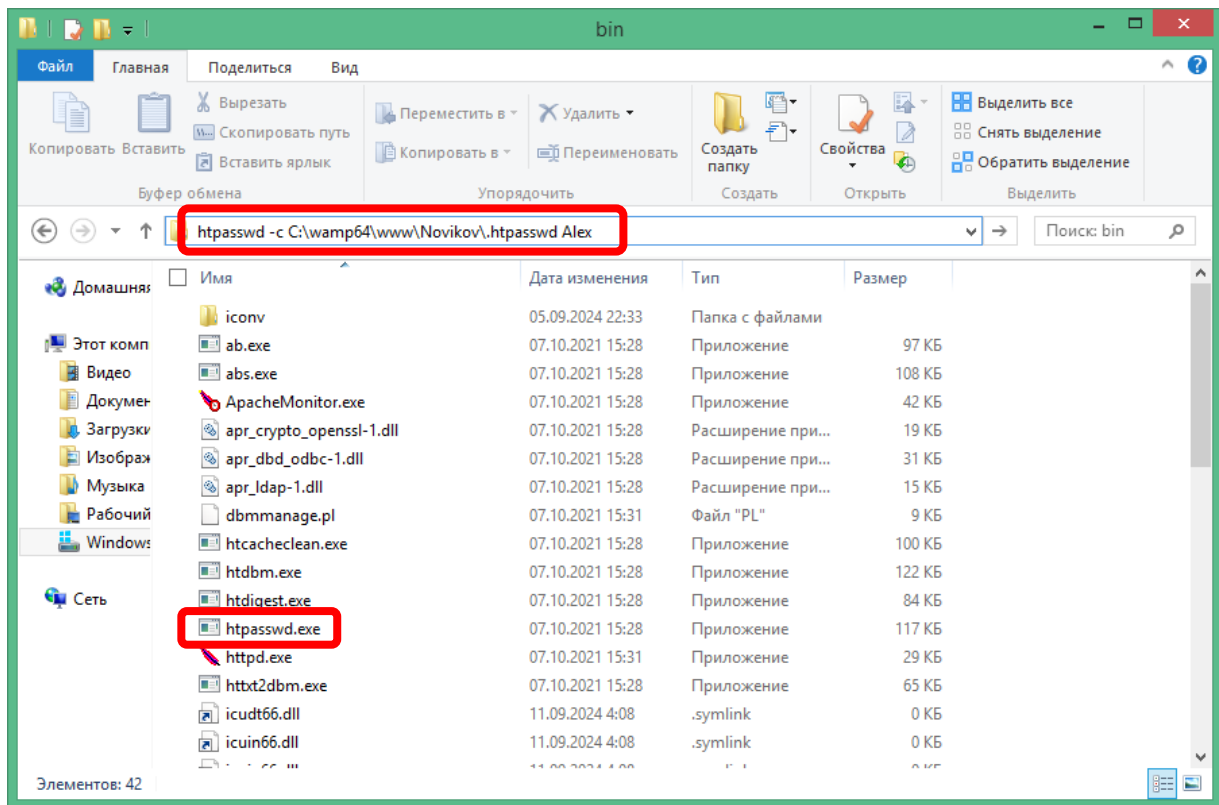


Рисунок 242 – Запуск программы htpasswd

В появившемся окне необходимо дважды ввести один и тот же пароль (рис. 243).

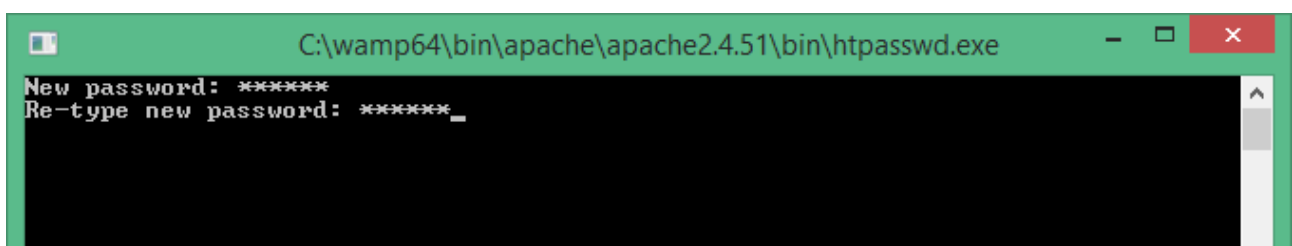


Рисунок 243 – Создание пароля

В результате будет создан файл «C:\wamp64\www\Novikov\htpasswd».

7.12. *Доступ к серверу с других устройств

Ранее мы подключались из браузера к **Apache**-серверу (**WAMP**-серверу), расположенному на том же ПК. При этом использовался адрес «**localhost**» или «**127.0.0.1**». Далее будет рассказано, как подключиться к серверу с других устройств, расположенных в одной Локальной сети с сервером. В наиболее

классическом случае речь идет про то, что ПК с установленным **WAMP**-сервером и второй ПК (или смартфон) подключены к одному роутеру (не важно по Wi-Fi или кабелем).

Для подключения к серверу с другого устройства требуется:

- 1) узнать текущий **IP**-адрес сервера (в примере это будет «**192.168.0.120**»);
- 2) настроить **Брандмауэр** Windows (и, возможно, Firewall и Антивирус) на сервере;
- 3) настроить **Apache**-сервер;
- 4) также рекомендуется настроить **статический** локальный **IP**-адрес для сервера.

1. IP-адрес сервера

Для того чтобы узнать текущий **локальный IP**-адрес (**IPv4**-адрес) компьютера, существует несколько способов. Рассмотрим основные из них.

1. Нажать комбинацию клавиш «**Windows+R**», ввести «**cmd**» (рис. 244), а в появившемся окне ввести команду «**ipconfig**» (рис. 245).

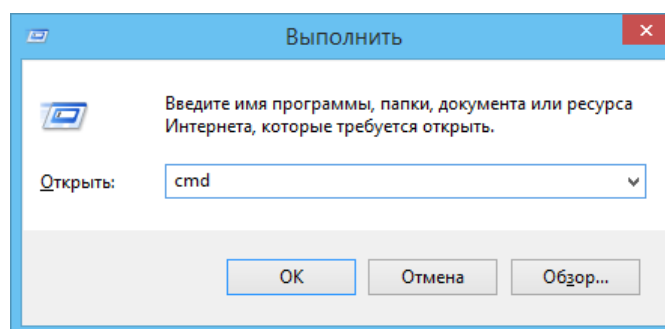


Рисунок 244 – Вызов «cmd»

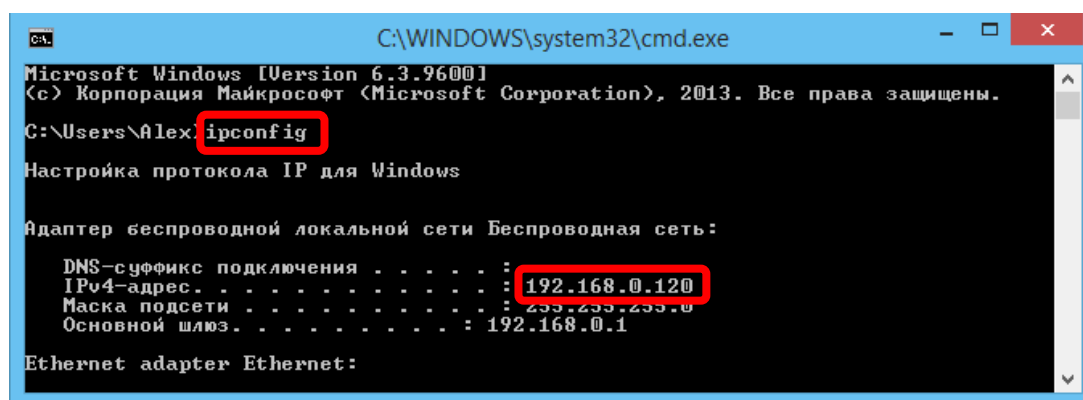


Рисунок 245 – Результаты команды «ipconfig»

2. Нажать комбинацию клавиш «**Windows+X**» и выбрать пункт «**Сетевые подключения**» (рис. 246). Дважды кликнуть мышкой по требуемому сетевому подключению и в открывшемся окне нажать кнопку

«Сведения». В появившемся окне будет отображен текущий **IP-адрес** (рис. 247).

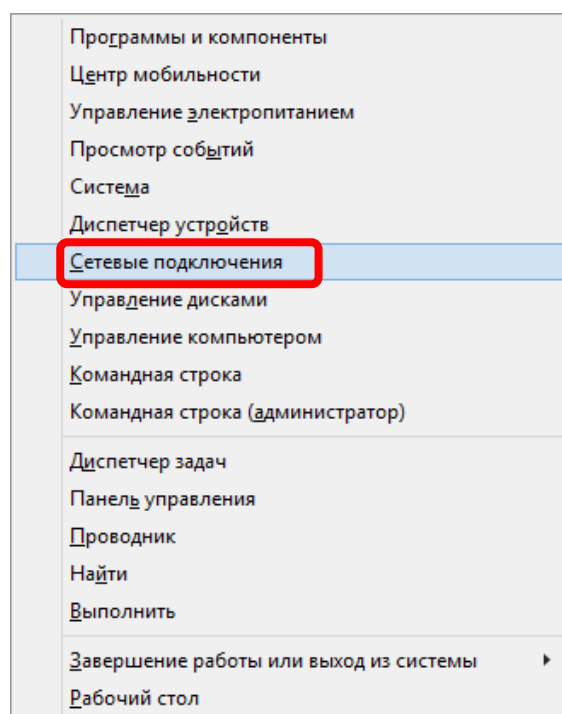


Рисунок 246 – Меню «Windows+X»

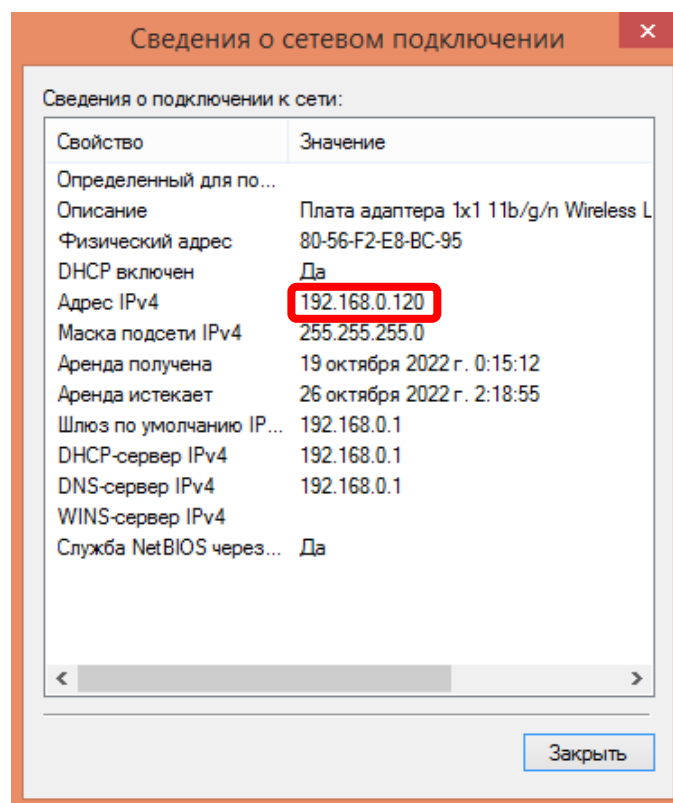


Рисунок 247 – Сведения о сетевом подключении

3. В углу у часов нажать правой кнопкой мышки на значок сети и выбрать «**Центр управления сетями и общим доступом**» (рис. 248). После чего выбрать «**Изменение параметров адаптера**» (рис. 249) и далее, как и в предыдущем способе, дважды кликнуть мышкой по требуемому сетевому подключению и в открывшемся окне нажать кнопку «**Сведения**». В появившемся окне будет отображен текущий IP-адрес (см. рис. 247).

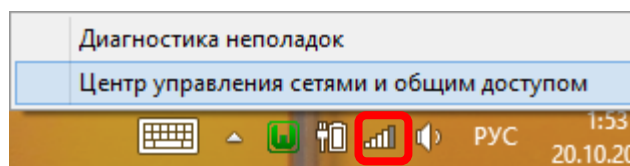


Рисунок 248 – Вызов Центра управления сетями

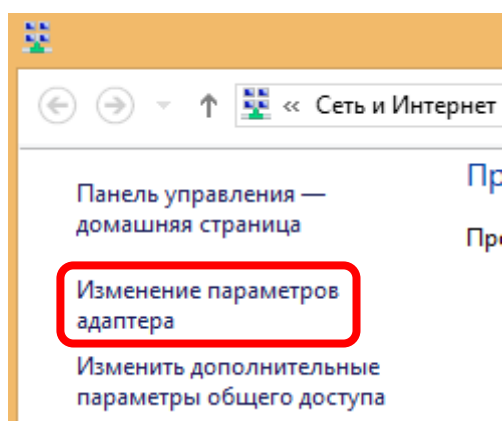


Рисунок 249 – Изменение параметров адаптера

Зайдем по найденному адресу через браузер (пока с того же компьютера, для которого мы определили этот адрес), и убедимся, что он работает (рис. 250). Также обратим внимание на то, что теперь браузер предупреждает нас, что подключение «Не защищено».

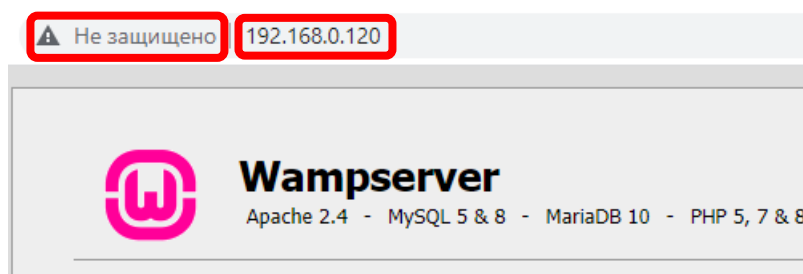


Рисунок 250 – Доступ по адресу 192.168.0.120

2. Настройка Брандмауэра

Попробуем зайти по этому же адресу (192.168.0.120) с любого другого устройства в этой Локальной сети, например со смартфона, подключенного по Wi-Fi к тому же роутеру. Практически наверняка вначале мы получим ошибку «Не удастся получить доступ к сайту» (рис. 251).

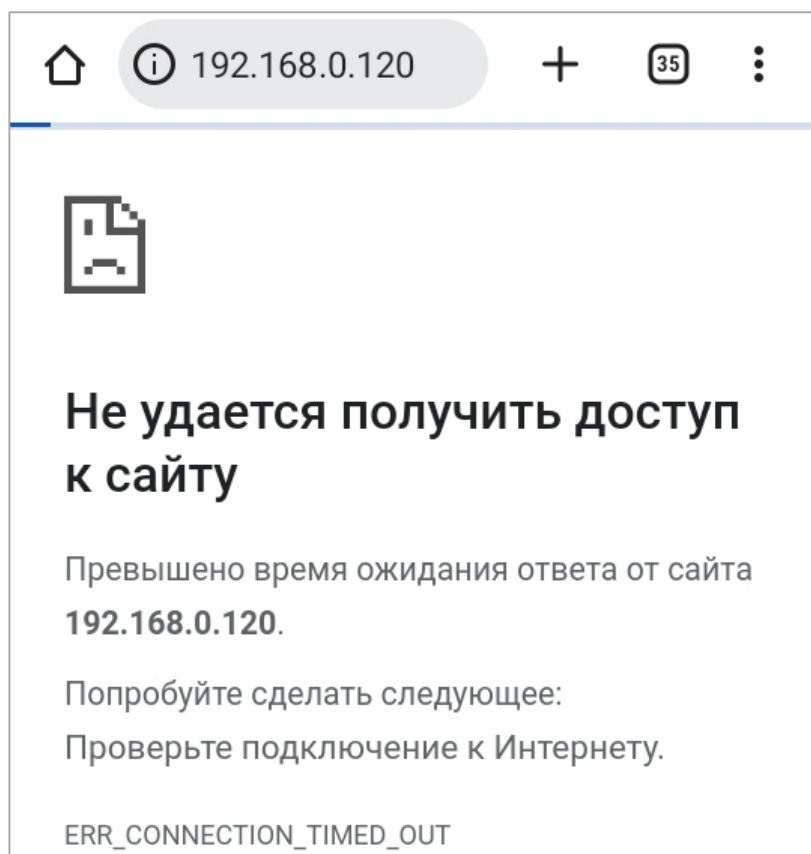


Рисунок 251 – Неудачное подключение со смартфона

Это означает, что соединение блокирует Брандмауэр Windows (и, возможно, Firewall и Антивирус). Для отключения Брандмауэра Windows необходимо перейти в «Панель управления\Система и безопасность\Брандмауэр Защитника Windows» (рис. 252).

!!! Стоит обратить внимание, что отключение Брандмауэра (Firewall-a, Антивируса) здесь осуществляется лишь для **демонстрации** возможности подключения к серверу. Не следует применять отключение Брандмауэра (Firewall-a, Антивируса) на постоянной основе, в дальнейшем необходимо настроить его таким образом, чтобы он разрешал работу Apache-сервера, но не отключал другие запреты.

Если все запреты Брандмауэра (Firewall-a, Антивируса) отключены правильно, то теперь заходя по адресу 192.168.0.120 мы получим сообщение об ошибке «**403 Forbidden**» – «Доступ запрещен» (рис. 253).

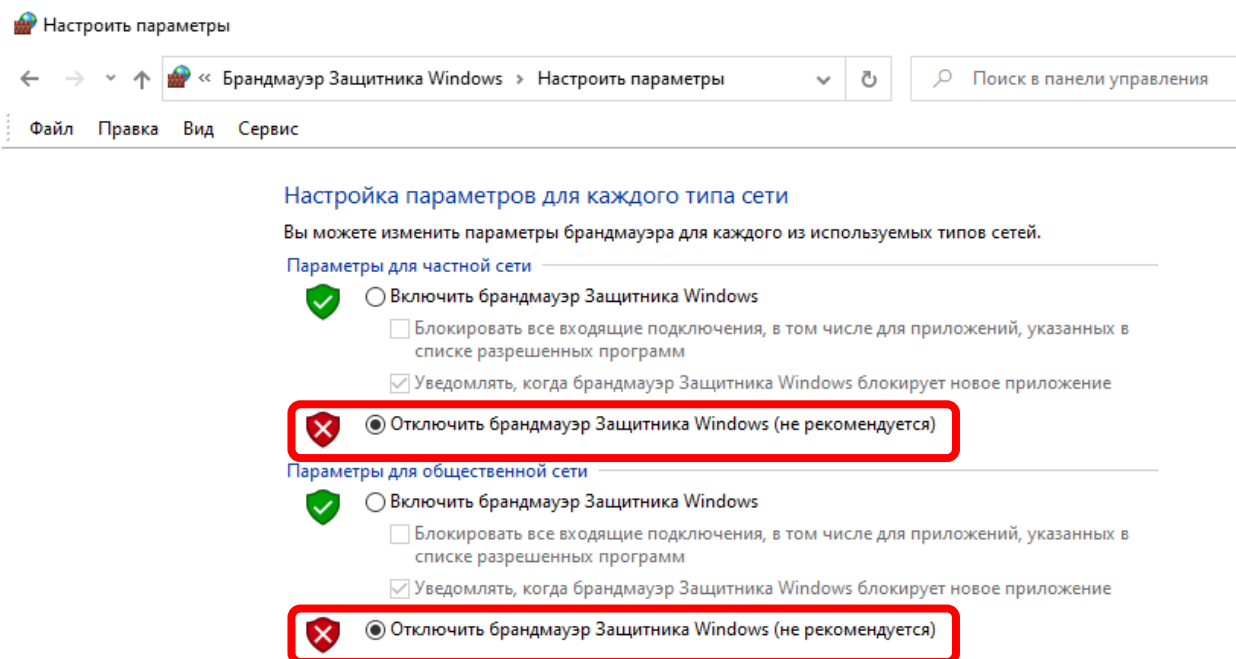


Рисунок 252 – Отключение Брандмауэра Windows



Рисунок 253 – Запрет доступа Apache-сервером

Это сообщение об ошибке «**403 Forbidden**» («Доступ запрещен») означает, что связь с Apache-сервером есть, но дальнейшая загрузка страницы блокируется самим сервером. *Подробнее про Коды ошибок см. раздел 7.9 «**Коды ошибок HTTP (404 и др.)».*

3. Настройка Apache-сервера

Когда мы наконец-то получили ответ от **Apache**-сервера, пусть и в виде ошибки «**403 Forbidden**» («Доступ запрещен»), пришло время настроить и его.

Откроем с помощью блокнота файл конфигурации, расположенный по адресу «C:\wamp64\bin\apache\apache2.4.51\conf\extra\httpd-vhosts.conf», и исправим в нем соответствующую строчку на «**Require all granted**» (рис. 254). После внесения изменений может потребоваться перезагрузить **WAMP**-сервер.

Чтобы обратно запретить доступ по сети, необходимо исправить значение на «**Require local**».

```
1 # Virtual Hosts
2 #
3 <VirtualHost *:80>
4     ServerName localhost
5     ServerAlias localhost
6     DocumentRoot "${INSTALL_DIR}/www"
7     <Directory "${INSTALL_DIR}/www/">
8         Options +Indexes +Includes +FollowSymLinks +MultiViews
9         AllowOverride All
10        Require all granted
11    </Directory>
12 </VirtualHost>
```

Рисунок 254 – Разрешение доступа к Apache-серверу

Теперь, заходя по адресу 192.168.0.120, мы получим доступ к требуемой странице (рис. 255).

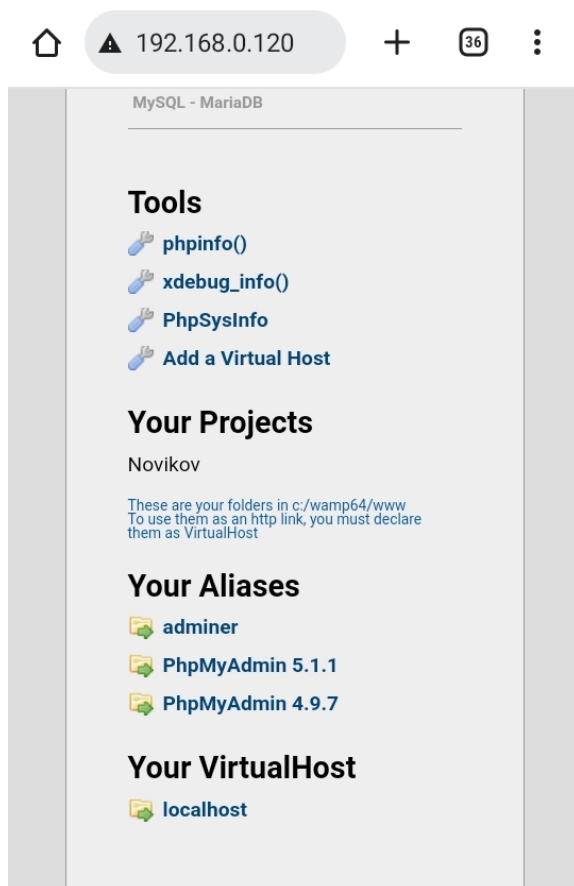


Рисунок 255 – Удачное подключение со смартфона

Если мы хотим иметь доступ и к **PhpMyAdmin**, то необходимо аналогично в файле «C:\wamp64\alias\phpmyadmin.conf» исправить строчку на «**Require all granted**» (рис. 256).

```

1 Alias /phpmyadmin "c:/wamp64/apps/phpmyadmin5.1.1/"
2
3 <Directory "c:/wamp64/apps/phpmyadmin5.1.1/">
4     options +Indexes +FollowSymLinks +MultiViews
5     AllowOverride all
6     Require all granted
7
8 # To import big file you can increase values
9     php_admin_value upload_max_filesize 128M
10    php_admin_value post_max_size 128M
11    php_admin_value max_execution_time 360
12    php_admin_value max_input_time 360
13 </Directory>

```

Рисунок 256 – Разрешение доступа к PhpMyAdmin

Стоит обратить внимание, что мы настроили доступ только к одной из версий **PhpMyAdmin** (самой новой из установленных). Для настройки других версий **PhpMyAdmin** (если их установлено несколько) необходимо внести аналогичные изменения в файлы настройки, у которых в имени файла указана версия, например, «C:\wamp64\alias\phpmyadmin4.9.7.conf».

4. Настройка статического локального IP-адреса

Если ПК с установленным **WAMP**-сервером имеет динамический **IP**-адрес, т.е. адрес, который присваивается роутером, то такой адрес может периодически менять свое значение, что затруднит подключение к такому серверу с других устройств.

Для сервера имеет смысл применять статический **IP**-адрес. Для его настройки необходимо в свойствах сетевого соединения (рис. 257) прописать IP версии 4 (рис. 258).

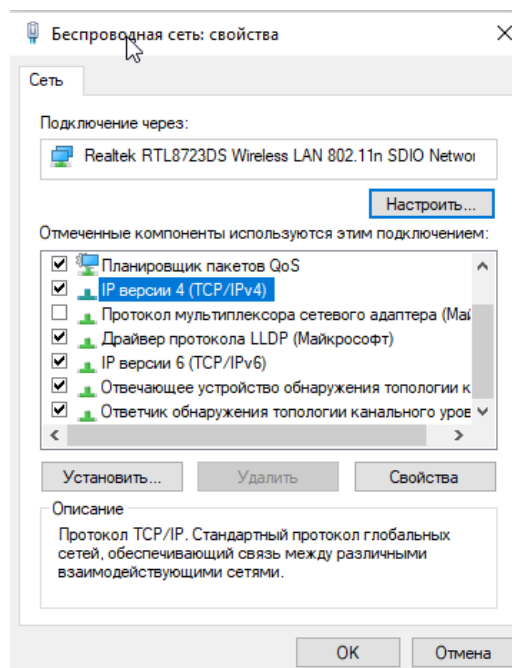


Рисунок 257 – Настройка IP-адреса

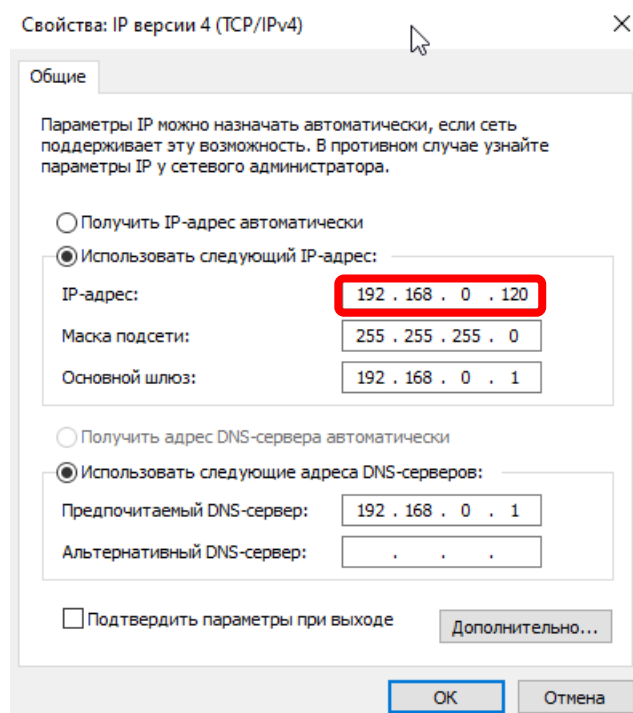


Рисунок 258 – Статический IP-адрес

7.13. **Шифрование и SSL-сертификат

SSL (Secure Sockets Layer – уровень защищённых сокетов) – криптографический протокол, который подразумевает более безопасную связь. Использует асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений. **SSL** считается устаревшим и должен быть исключён из работы в пользу **TLS**.

TLS (англ. Transport Layer Security – протокол защиты транспортного уровня) – фактически это новая версия **SSL**. В настоящее время под названием «**SSL**» обычно подразумевается, как сам **SSL**, так и **TLS**.

SSL изначально разработан компанией Netscape Communications для добавления протокола **HTTPS** в свой веб-браузер Netscape Navigator. Впоследствии на основании протокола **SSL 3.0** был разработан и принят стандарт **RFC**, получивший имя **TLS**.

HTTPS (HyperText Transfer Protocol Secure) – расширение протокола **HTTP** для поддержки шифрования в целях повышения безопасности. Данные в протоколе **HTTPS** передаются поверх криптографического протокола **TLS** (или устаревшего в 2015 году **SSL**). В отличие от **HTTP** с TCP-портом 80, для **HTTPS** по умолчанию используется TCP-порт **443**. Протокол разработан компанией Netscape Communications для браузера Netscape Navigator в 1994 году.

OpenSSL – программа для генерации **SSL**-сертификатов.

Для включения функции шифрования необходимо выполнить следующие два действия:

- 1) сгенерировать сертификат (пару файлов сертификат и ключ):

- для публичного сервера, у которого имеется домен в интернете, для получения **SSL**-сертификата необходимо обратиться в один из центров сертификации (это платная услуга);
- для частного сервера генерируется «самоподписанный сертификат». В составе **WAMP**-сервера для этого уже имеется программа **OpenSSL**, она расположена по адресу:
«C:\wamp64\bin\apache\apache2.4.51\bin\openssl.exe».

2) установить сертификат и настроить **WAMP**-сервер (**Apache**-сервер).

Для установки сертификата также может понадобиться включить модуль для работы с сертификатами «ssl_module» (рис. 259). Для этого необходимо нажать левую клавишу мышки на значке WAMP-сервера в трее, после чего выбрать «Apache» → «Apache modules» («Модули Apache»).

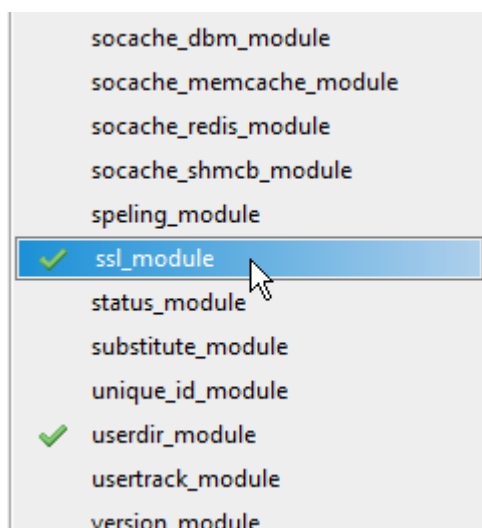


Рисунок 259 – Включение модуля SSL

7.14. **Доступ к серверу из глобальной сети

Итак, мы настроили **SSL**-шифрование, и наши пароли теперь в безопасности, пришло время задуматься о подключении к серверу через сеть Интернет.

Обозначим основные направления, которые необходимо изучить для достижения этой цели:

- «**Проброс портов**» («перенаправление портов», «переедресация портов») – т.е. настройка роутера на поступление запросов извне (из Интернета на сервер в Локальной сети). По умолчанию используются следующие порты:
 - **HTTP** – порт **80**;
 - **HTTPS** – порт **443**;
 - **MySQL/MariaDB** - порт **3306** (3307, 3308);

- **FTP** – порт **21** (а также порт 20 и некоторые из портов динамического диапазона 49152–65535, например для QNAP NAS диапазон портов *пассивного FTP-режима* 55536–56559);
- «**Статический IP-адрес**» в глобальной сети (выдается провайдером);
- либо «**Динамический DNS**» («**DDNS**») для сервера без статического (глобального) IP-адреса, т.е. с динамическим IP-адресом (хотя DDNS может применяться как к динамическому, так и к статическому IP-адресу, позволяя использовать вместо трудно запоминаемого набора цифр понятное символьное имя вида «MyName.ddns.net»). Функция DDNS уже имеется на многих роутерах, остается только зарегистрироваться в одном из сервисов DDNS и правильно настроить роутер.

!!! Для получения доступа через Интернет вначале должен быть настроен доступ по Локальной сети (см. раздел 7.12).

7.15. *Автозапуск WAMP-сервера

По умолчанию для работы с **WAMP**-сервером его нужно запустить вручную (ярлык на рабочем столе). Автоматический запуск **WAMP**-сервера (**Apache**-сервера и **MySQL**-сервера) может быть полезен, например, после восстановления электропитания (после аварийного отключения). При этом если сервер настроен на работу в глобальной сети (Интернет) или на сохранение в БД показаний датчика, поступающих с контроллера, то такой автоматический перезапуск сервера просто необходим.

Настройку автозагрузки программ можно осуществить при помощи Планировщика заданий Windows. Для запуска **Планировщика заданий** необходимо нажать правую клавишу мышки на значке «Мой компьютер» («Этот компьютер») и выбрать пункт меню «**Управление**» (рис. 260). Другой способ попасть в Планировщик заданий – это запустить «Панель управления» и в ней выбрать «Система и безопасность» → «Администрирование» → «Расписание выполнения задач».

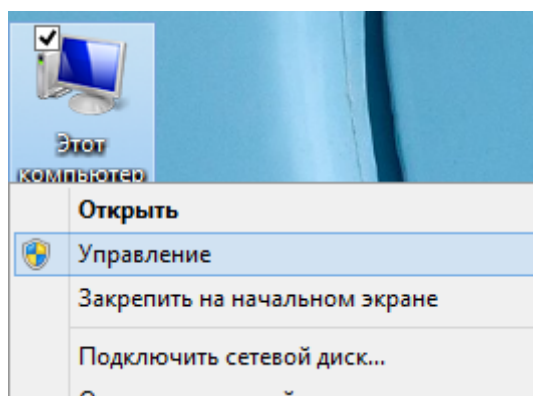


Рисунок 260 – Вызов Управления компьютером

Далее необходимо выбрать «Планировщик заданий» и нажать кнопку «Создать задачу» (рис. 261).

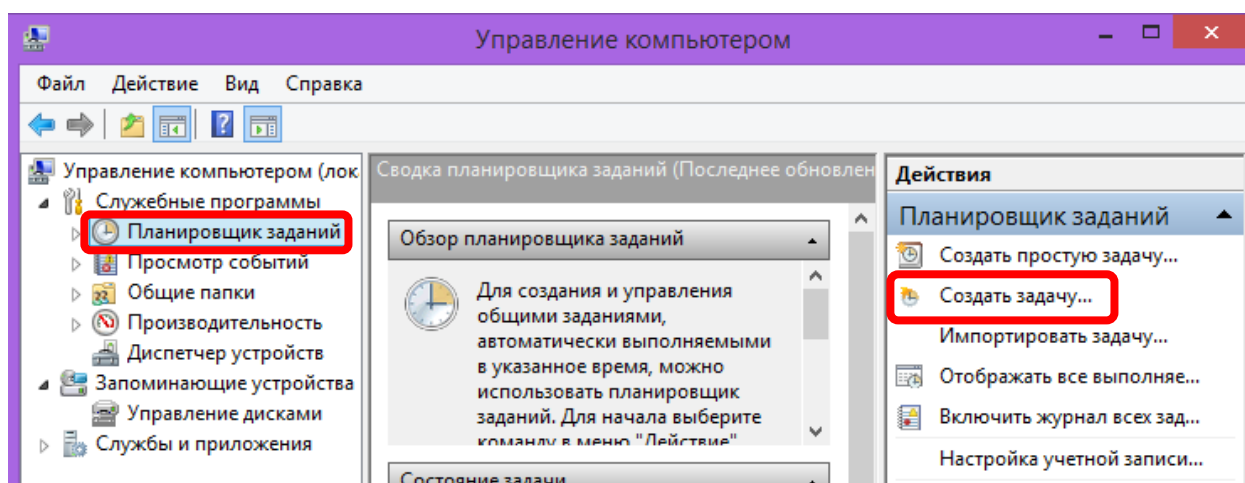


Рисунок 261 – Создание задачи в планировщике

В открывшемся окне необходимо указать «Имя» создаваемой задачи и поставить галочку «Выполнить с наивысшими правами» (рис. 262).

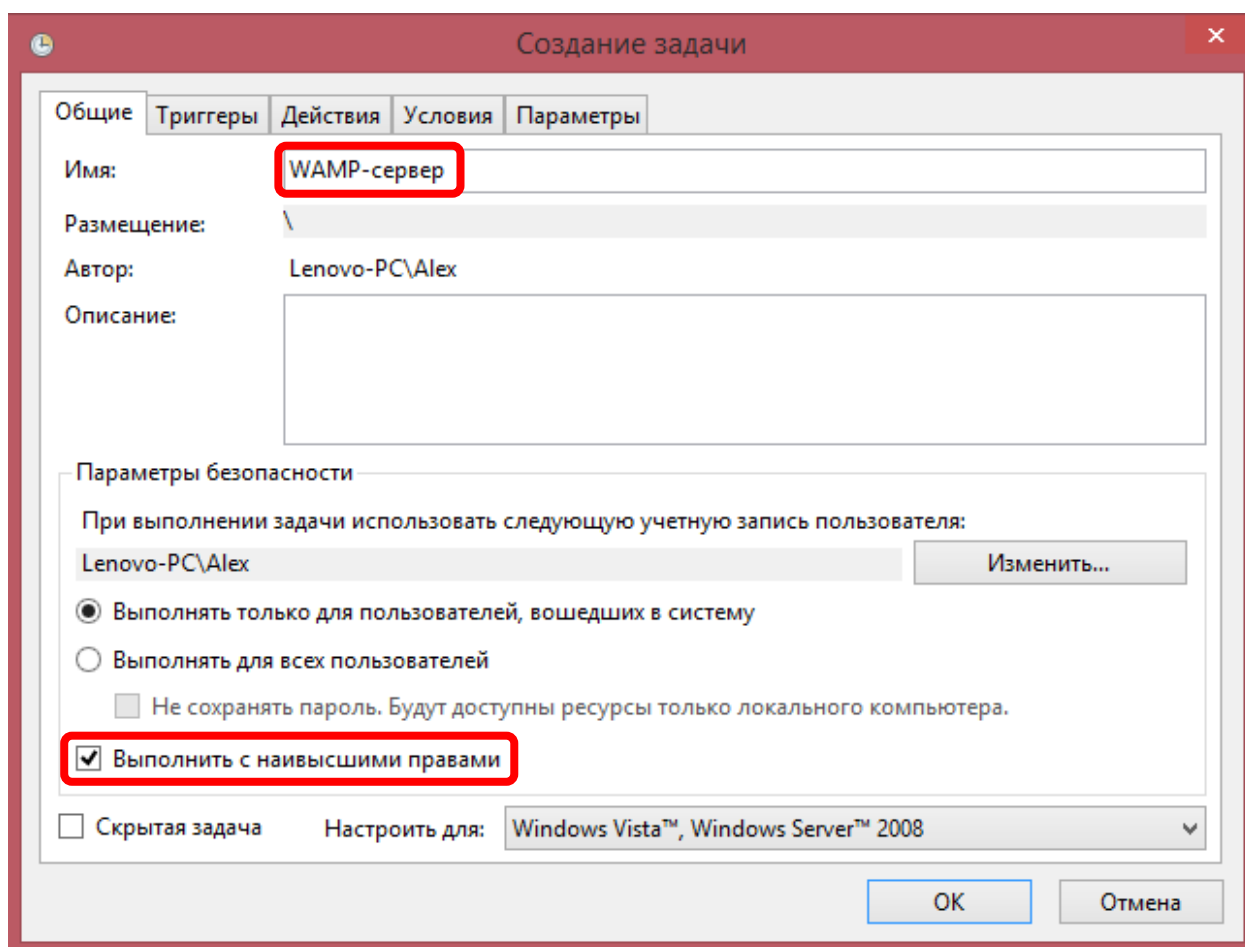


Рисунок 262 – Настройка создаваемой задачи

Далее переходим на вкладку «Триггеры» (рис. 263) и указываем условия запуска программы. Для этого необходимо нажать кнопку «Создать», и в появившемся окне указать «При входе в систему» и «Любой пользователь».

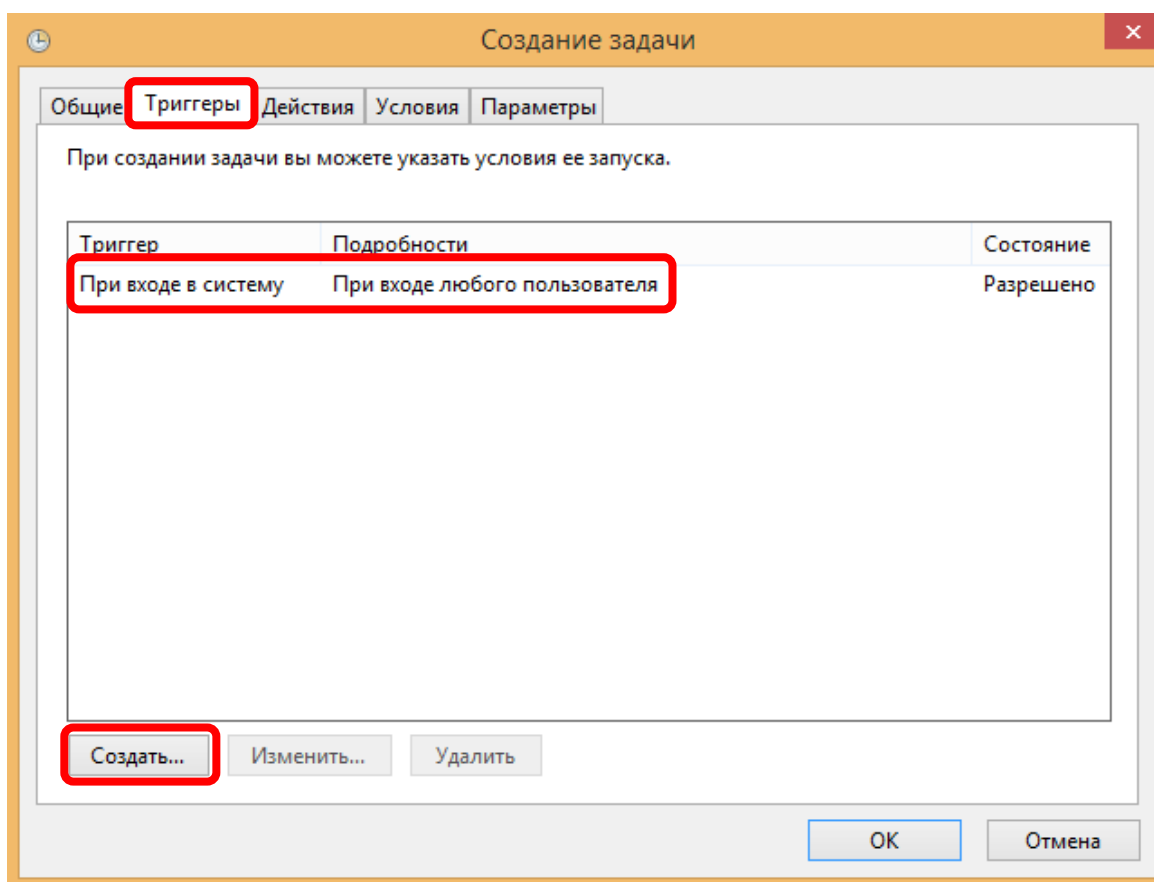


Рисунок 263 – Настройка условий запуска задачи

На вкладке «Действия» (рис. 264) указываем какую программу требуется запустить. Для этого необходимо нажать кнопку «Создать», в появившемся окне указать «Запуск программы» и, нажав кнопку «Обзор», выбрать адрес до WAMP-менеджера (при установке WAMP-сервера в папку по умолчанию, это адрес «C:\wamp64\wampmanager.exe»).

На вкладках «Условия» (рис. 265) и «Параметры» (рис. 266) необходимо снять галочки, как показано на соответствующих рисунках.

После завершения создания задачи необходимо **перезагрузить компьютер** и убедиться в появлении иконки WAMP-сервера у часов (рис. 267).

Для просмотра и изменения (редактирования, отключения, удаления) созданной задачи необходимо зайти в Планировщик заданий и выбрать «Библиотека планировщика заданий» (рис. 268), после чего в списке (по алфавиту) найти созданную ранее задачу.

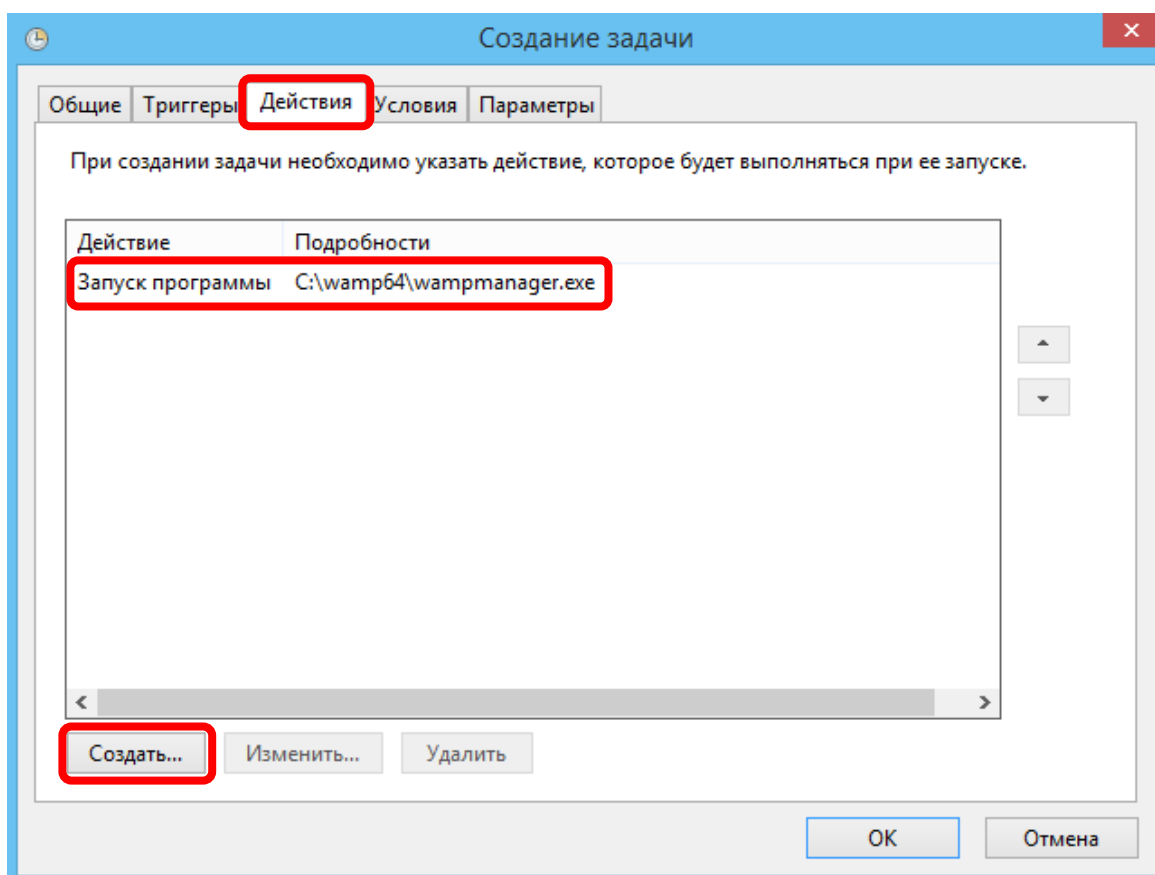


Рисунок 264 – Настройка действия, выполняемого задачей

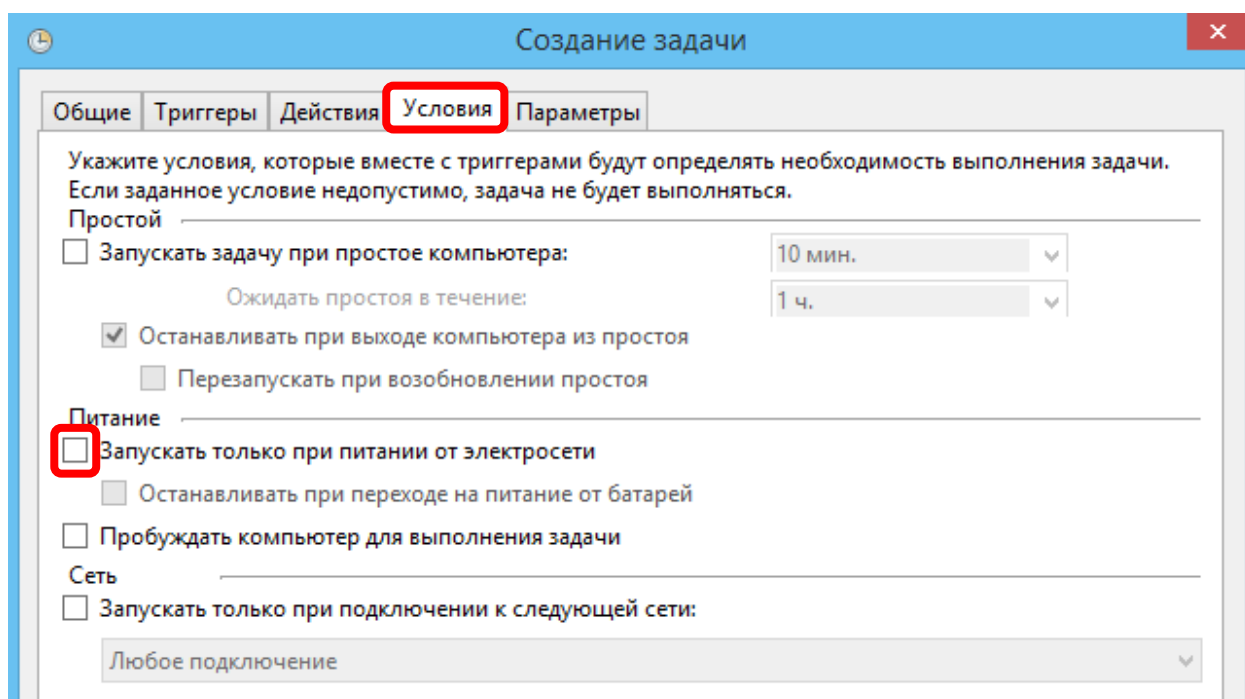


Рисунок 265 – Настройка дополнительных условий

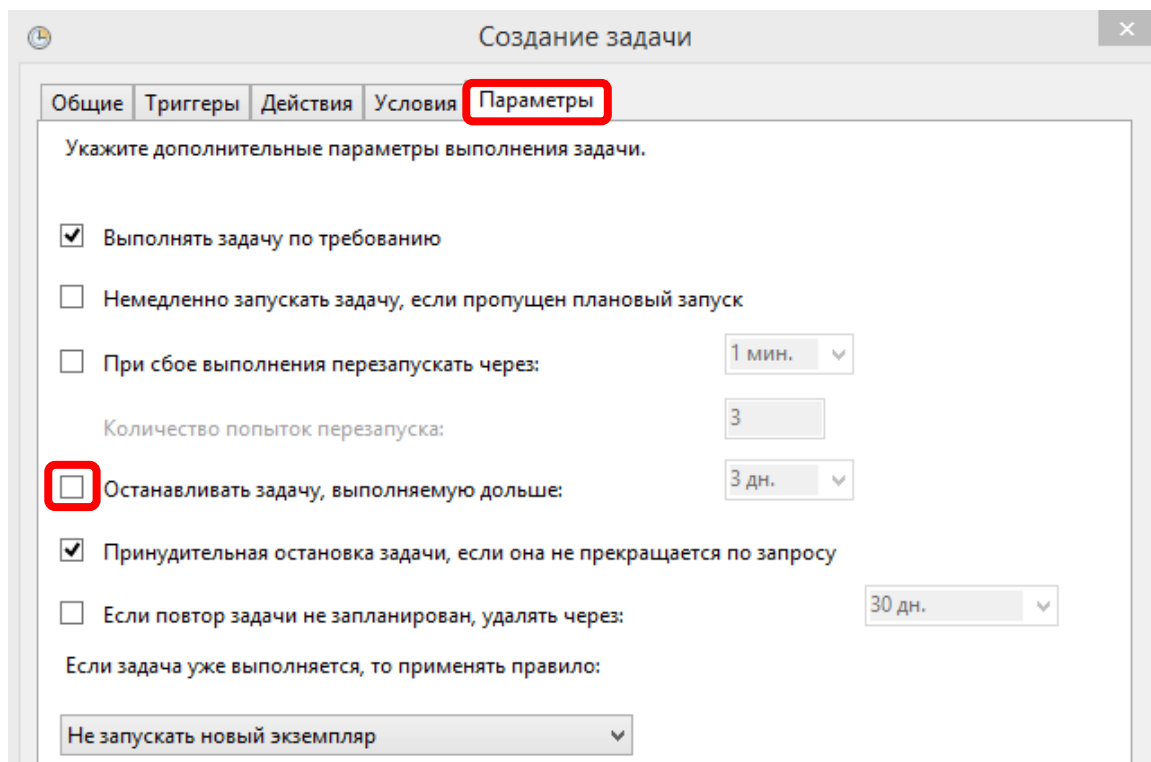


Рисунок 266 – Настройка дополнительных параметров



Рисунок 267 – Иконка WAMP-сервера у часов

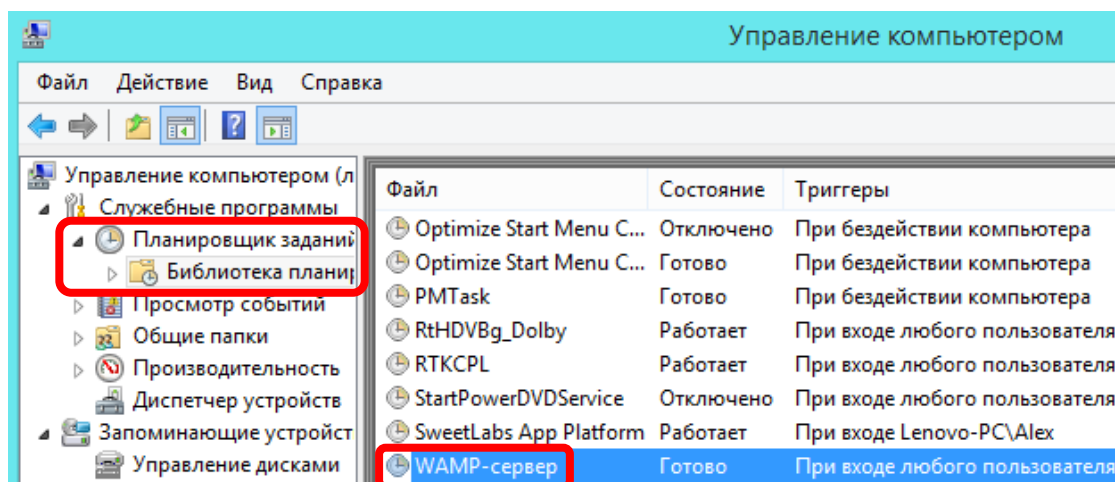


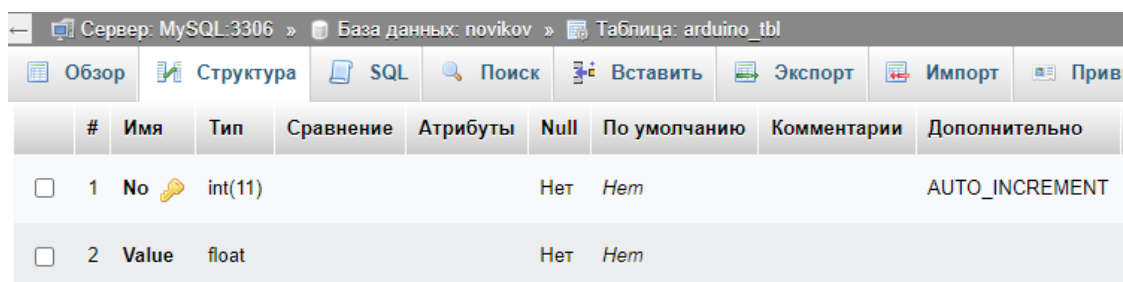
Рисунок 268 – Просмотр созданной задачи в Планировщике заданий

7.16. **Arduino и MySQL

Будем передавать данные с контроллера **Arduino** в базу данных **MySQL**. Для начала создадим саму таблицу при помощи **PhpMyAdmin**, а также «прослойку» на **PHP**, через которую **Arduino** будет связываться с базой данных.

1. Сервер

При помощи **PhpMyAdmin** создадим таблицу «**arduino_tbl**», структура которой представлена на рисунке 269.



	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
☐	1	No	int(11)			Нет	Нет		AUTO_INCREMENT
☐	2	Value	float			Нет	Нет		

Рисунок 269 – Создание таблицы

Далее создадим файл «**Arduino.php**», со следующим кодом:

```
<?php
//Значение, которое нужно добавить в базу
//(например: localhost/Novikov/Arduino.php?val=7.5 )
$val = $_GET['val'];

//Подключиться к базе данных
$connect = mysqli_connect("localhost", "root", "", "Novikov");
//Завершить работу в случае ошибки
if ($connect == false) {
    print("Невозможно подключиться к MySQL");
    exit;
}
//Установить кодировку
mysqli_set_charset($connect, 'utf8');

//Добавить значение в базу данных
$sql = "INSERT INTO arduino_tbl VALUES (null, $val);";
$result = mysqli_query($connect, $sql);
//Завершить работу в случае ошибки
if ($result == false) {
    print("Ошибка при выполнении запроса<br>$sql");
    exit;
}

print("Ok!");

//Завершить соединение с базой данных
mysqli_close($connect);
?>
```

Теперь после каждого входа по адресу вида «**localhost/Novikov/Arduino.php?val=7**» в таблицу «**arduino_tbl**» (рис. 270) будет добавляться новое значение, указанное в адресной строке.

!!! В дальнейшем необходимо снабдить таблицу колонкой с метками времени, чтобы знать точные дату и время сохранения каждого значения.

No	Value
1	1
2	2
3	3.3
4	9
5	9.1
6	7

Рисунок 270 – Таблица arduino_tbl

В случае удачной записи значения в браузере отобразится «Ok!» (рис. 271).

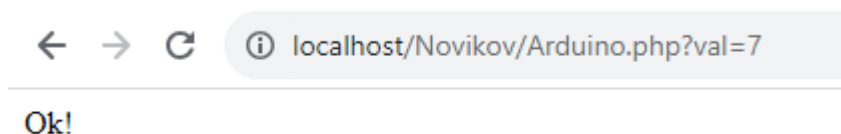


Рисунок 271 – Успешная запись значения

2. Клиент (Arduino)

Далее научим контроллер самостоятельно заходить по данному адресу. Для этого в **Arduino IDE** напишем следующий код, который необходимо загрузить в контроллер **Arduino**:

```
#include <SPI.h>
#include <Ethernet.h>
EthernetClient client;

//Адрес сервера, к которому осуществляется подключение.
char server[] = "192.168.0.132";
String address = "/Novikov/Arduino.php?val=";

//Настройка адреса текущего устройства.
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
```

```

void setup() {
  Serial.begin(9600);
  #define LED 8
  pinMode(LED, OUTPUT);

  //Запускаем Ethernet-соединение.
  //IP-адрес получаем по DHCP.
  Serial.println("Initialize Ethernet with DHCP:");
  if (Ethernet.begin(mac) == 0) {
    Serial.println("  Error DHCP");
    while (true) {delay(1);}
  } else {
    Serial.print("  DHCP assigned IP ");
    Serial.println(Ethernet.localIP());
  }

  delay(1000);
}

void loop() {
  //В случае успешной передачи, светодиод мигает.
  //В случае ошибки, светодиод горит постоянно.
  digitalWrite(LED, HIGH);

  //Подключение к серверу (порт 80).
  Serial.print("Connecting to ");
  Serial.print(server);
  Serial.println("... ");
  if (!client.connect(server, 80)) {
    //Ошибка подключения к серверу.
    Serial.println("  Error");
  }

  //В случае успешного подключения...
  if (client.connected()) {
    //Считываем текущее значение с аналогового датчика A0.
    int val = analogRead(A0);
    Serial.print("  val = ");
    Serial.println(val);

    //Выполняем HTTP-запрос.
    client.print("GET ");
    client.print(address);
    client.println(val);
    client.println();

    //Отключиться от сервера.
    client.stop();
    digitalWrite(LED, LOW);
  }

  delay(500);
}

```

Данный код каждые **0,5 сек.** замеряет показания датчика, подключенного к **A0**, и передает это значение при помощи **HTTP**-запроса. Далее программа «**Arduino.php**» помещает данное значение в базу данных.

Задействованное в проекте «железо»:

- 1) контроллер Arduino UNO;
- 2) модуль Ethernet Shield;
- 3) светодиод (LED), подключенный к D8;
- 4) аналоговый датчик, подключенный к A0 (измеряемый параметр может быть любым).

Контроллер подключается кабелем к роутеру и после включения питания получает свой **IP**-адрес по **DHCP**.

!!! Контроллер нужно подключать именно к **роутеру**, т.к. при прямом соединении к ПК, он не сможет получить адрес по **DHCP**.

!!! Нужно не забывать отключать **Брандмауэр** перед подключением контроллера к базе данных.

Информация, отображаемая в «Мониторе порта» **Arduino IDE** во время работы контроллера, представлена на рисунке 272.

3. Отображение результатов

PhpMyAdmin позволяет отображать данные не только в виде таблиц, но и в виде **графика**. Для этого необходимо нажать соответствующую кнопку (рис. 273).

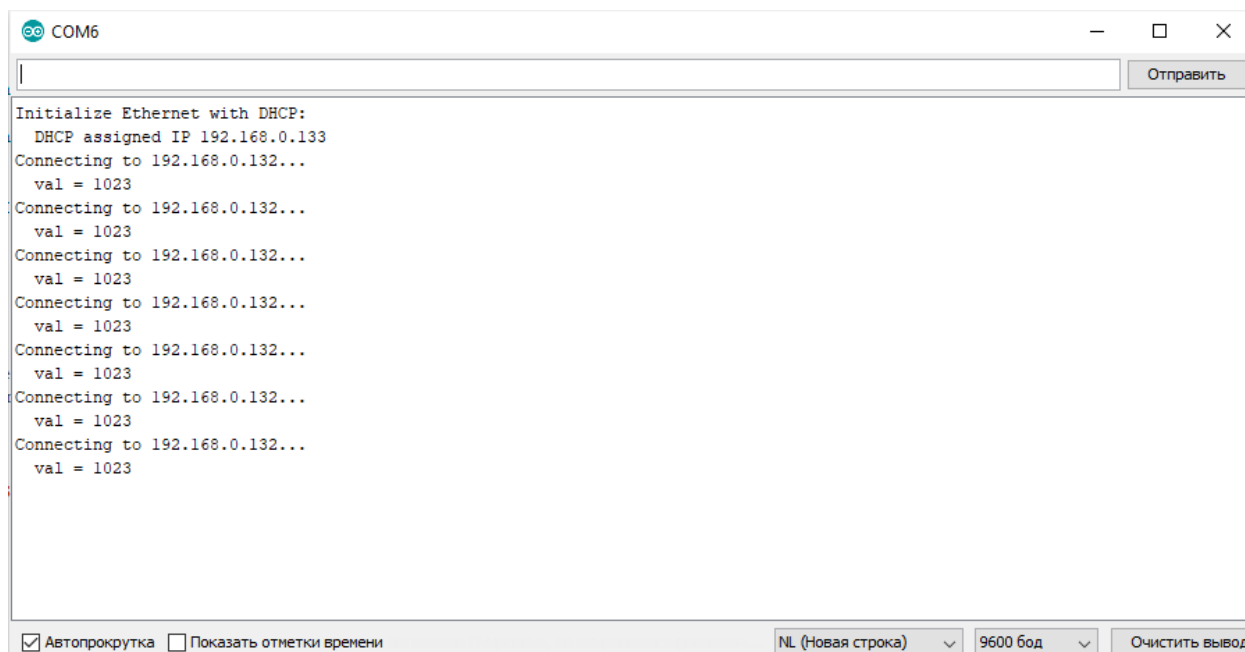


Рисунок 272 – Монитор порта

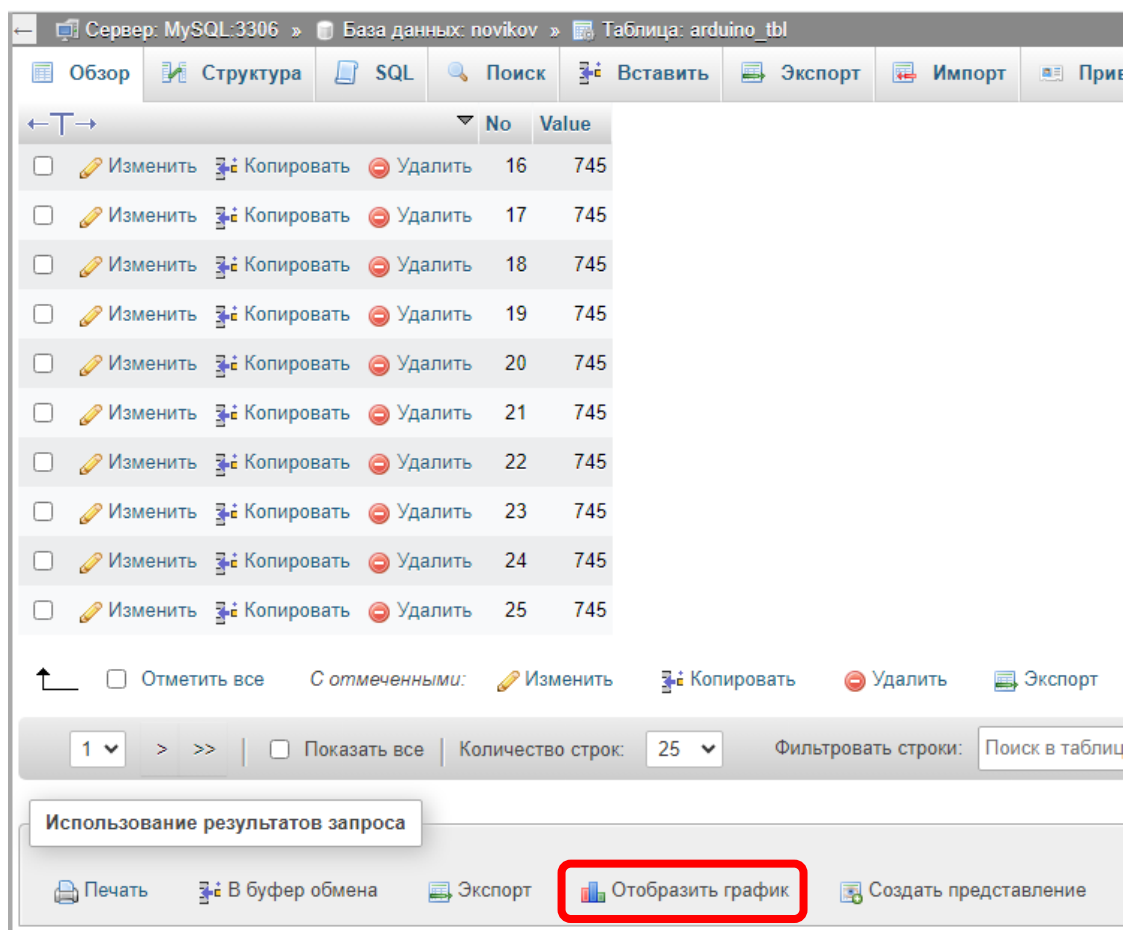


Рисунок 273 – Построение графика

Пример графика, построенного по собранным контроллером данным, представлен на рисунке 274.

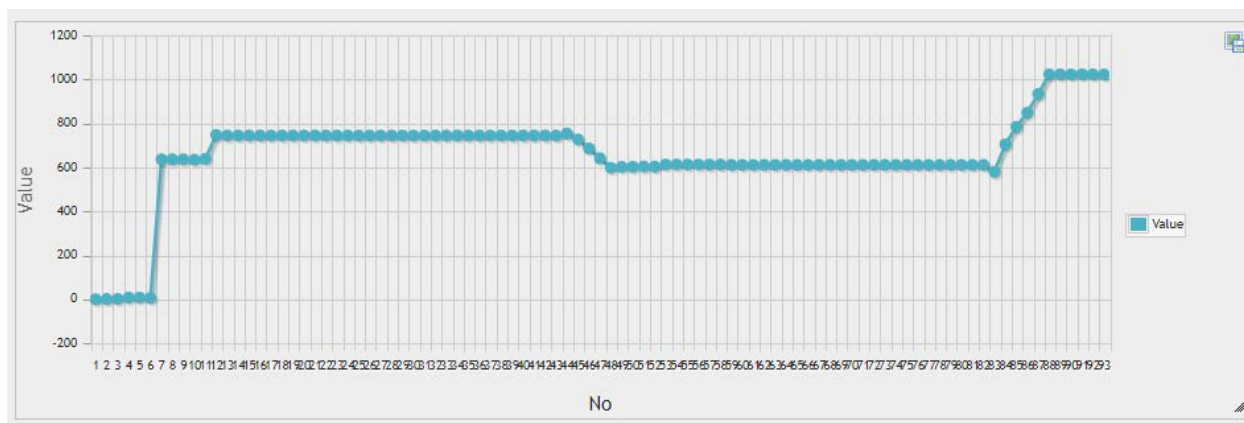


Рисунок 274 – График в PhpMyAdmin

Естественно, что нет смысла только сохранять значения в базу и нигде их не отображать. Поэтому желательно просматривать данные не только при помощи **PhpMyAdmin**. В дальнейшем имеет смысл создать отдельную **PHP**-страницу, которая отображает последнее (текущее) значение, а также позволяет просмотреть архив значений.

7.17. **Установка и настройка WAMP-сервера для компьютерного класса

Особенностью использования ПО в компьютерном классе является то, что работа за компьютерами осуществляется из учетных записей **без прав администратора**. Но при этом для запуска WAMP-сервера права администратора необходимы.

Решением данной проблемы является настройка автоматического запуска WAMP-сервера с правами администратора, что можно осуществить при помощи Планировщика заданий Windows. При этом студент не будет иметь возможности изменять настройки WAMP-сервера (и даже не будет видеть его иконку у часов), но сможет осуществлять подключение к нему из клиентской программы (браузера), будет иметь доступ к базе данных MySQL через менеджер PhpMyAdmin.

1. Установка WAMP-сервера

Установка **WAMP**-сервера в основном производится с настройками по умолчанию. При этом стоит обратить внимание, что установка должна производиться в папку «**C:\wamp64**», т.е. только в корень диска (нельзя устанавливать данную программу в «**C:\Program Files ...**»!). Путь не должен содержать пробелов и других специальных знаков, а также русских букв. Установка возможна только от имени **Администратора**.

!!! Нельзя устанавливать **WAMP**-сервер поверх имеющейся версии! Если ранее в данную папку уже был установлен **WAMP**-сервер, то необходимо вначале полностью удалить предыдущую версию при помощи стандартной функции Windows «Удалить или изменить программу», а при необходимости (если папка осталась) также удалить папку «**C:\wamp64**» вручную.

!!! При установке на **Windows 8** (и более ранние версии) необходимо выбрать для установки **MySQL версии 5**. При установке на **Windows 10** и более новые версии необходимо выбрать для установки **MySQL версии 8** (рис. 275).

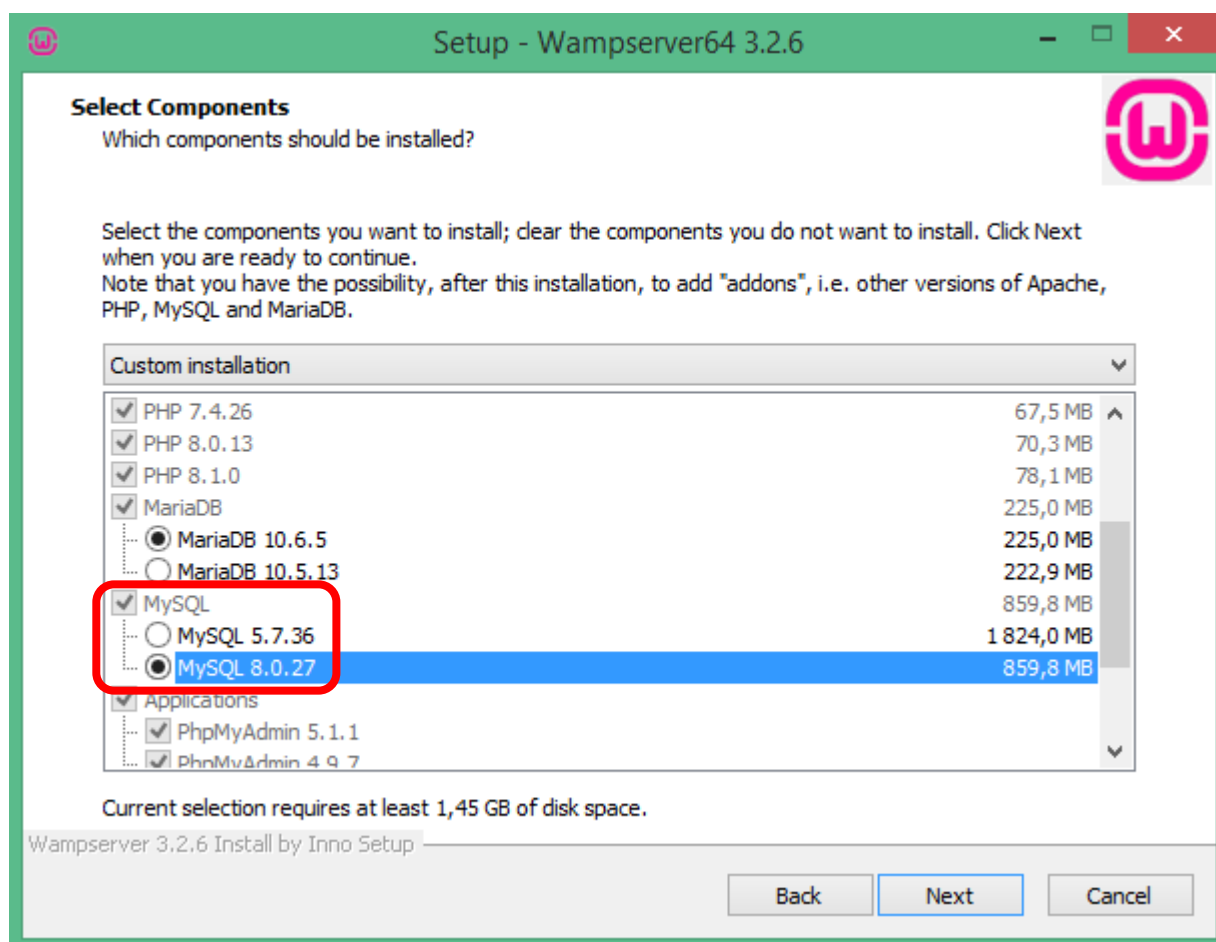


Рисунок 275 – Выбор версии MySQL для установки

2. Проверка работы MySQL-сервера

Для проверки правильности установки **WAMP**-сервера (и далее **MySQL**-сервера, входящего в его состав) необходимо проделать следующие действия:

— запустить появившийся на рабочем столе ярлык «**Wampserver64**»;

!!! Сервер стартует не мгновенно, после входа в учетную запись **необходимо подождать загрузки сервера не менее 1 минуты**. После запуска значок у часов должен стать зеленым (рис. 276).



Рисунок 276 – Значок WAMP-сервера у часов

— открыть браузер и ввести адрес **localhost**. Далее выбрать «**PhpMyAdmin**» (рис. 277);

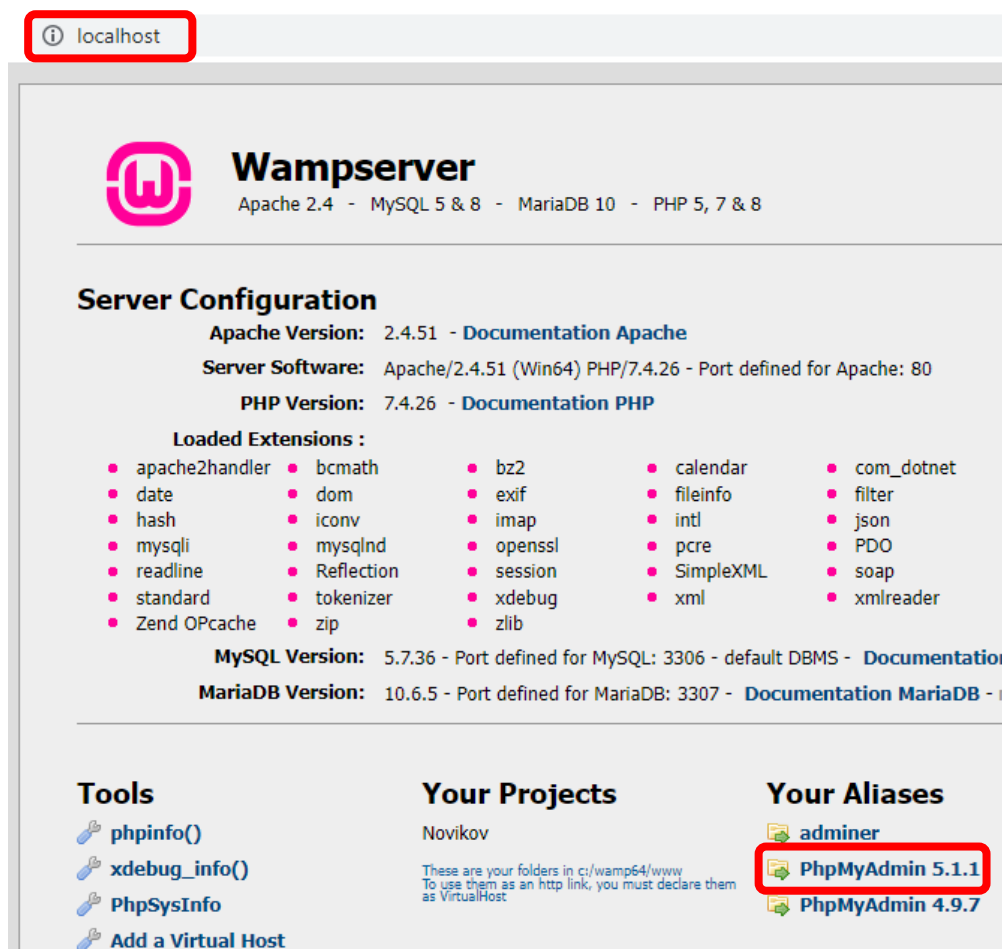


Рисунок 277 – Подключение к WAMP-серверу через браузер

— ввести имя пользователя **root** (без пароля) и подключиться к серверу **MySQL** (рис. 278).

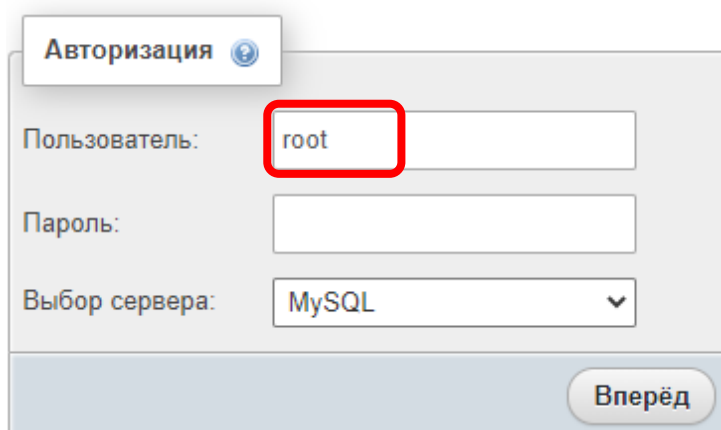


Рисунок 278 – Подключение к серверу MySQL

3. Настройка автоматического запуска WAMP-сервера

Если проверка работы **MySQL**-сервера прошла успешно, то далее необходимо настроить его автозапуск через Планировщик заданий.

Для запуска Планировщика заданий необходимо нажать правую клавишу мышки на значке «Мой компьютер» («Этот компьютер») и выбрать пункт меню «Управление» (рис. 279). Другой способ попасть в Планировщик заданий – это запустить «Панель управления» и в ней выбрать «Система и безопасность» → «Администрирование» → «Расписание выполнения задач».

Далее необходимо выбрать «Планировщик заданий» и нажать кнопку «Создать задачу» (рис. 280).

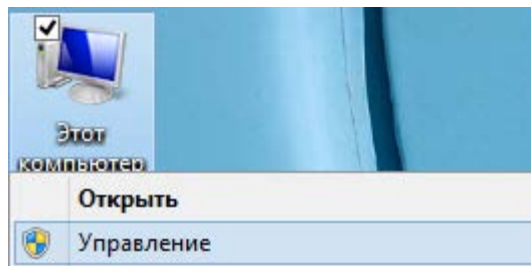


Рисунок 279 – Вызов Управления компьютером

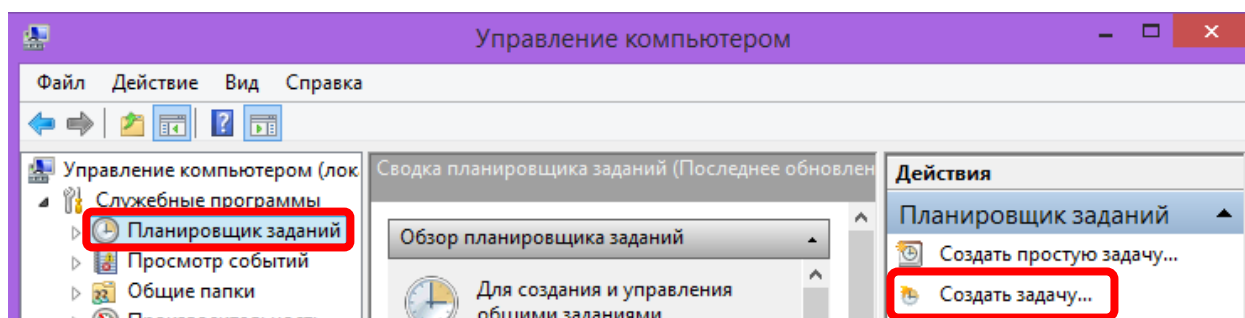


Рисунок 280 – Создание задачи в планировщике

В открывшемся окне произведем следующие настройки (рис. 281):

1. указываем «Имя» создаваемой задачи;
2. указываем описание создаваемой задачи (не обязательно);
3. выбираем пункт «Выполнить для всех пользователей»;
4. ставим галочку «Выполнить с наивысшими правами».

Далее переходим на вкладку «Триггеры» (рис. 282) и указываем условия запуска программы. Для этого необходимо нажать кнопку «Создать», и в появившемся окне (рис. 283) указать «При входе в систему» и «Любой пользователь».

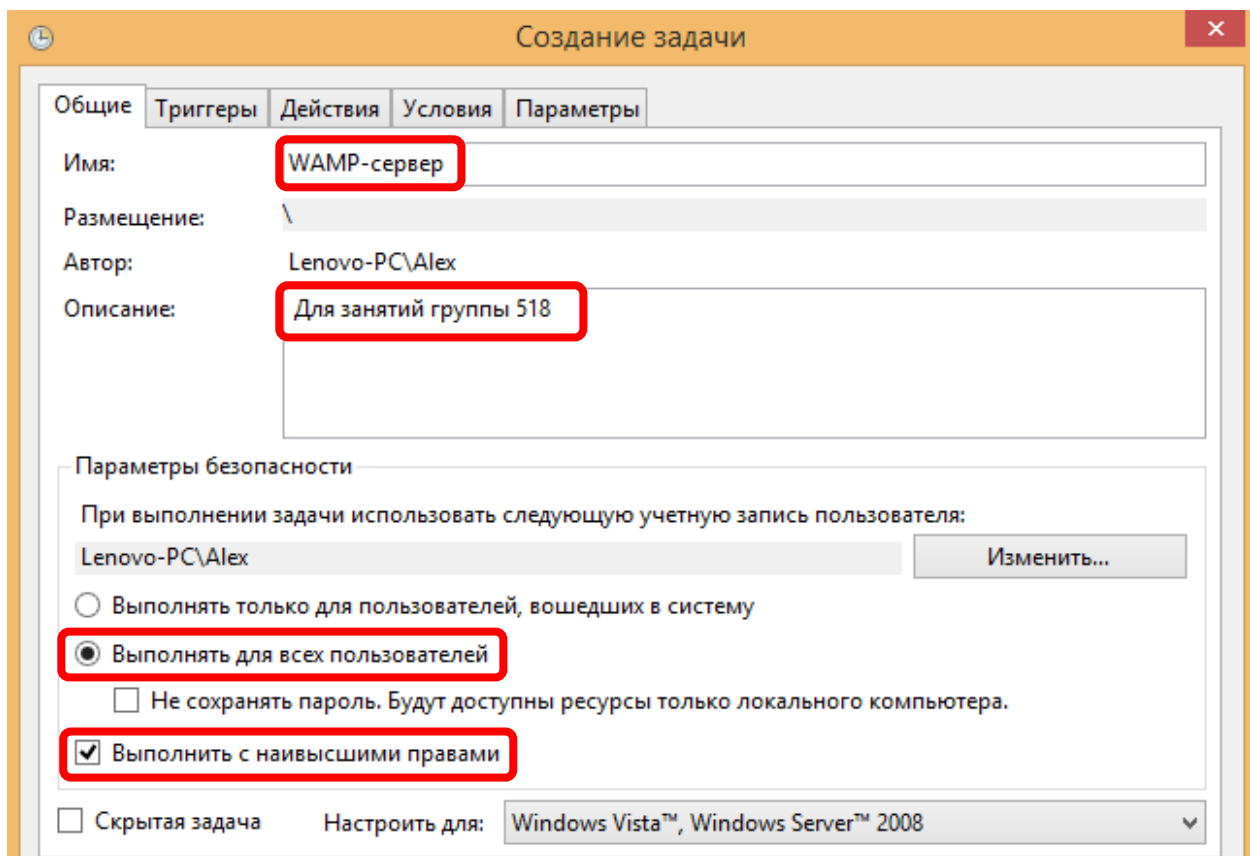


Рисунок 281 – Настройка создаваемой задачи

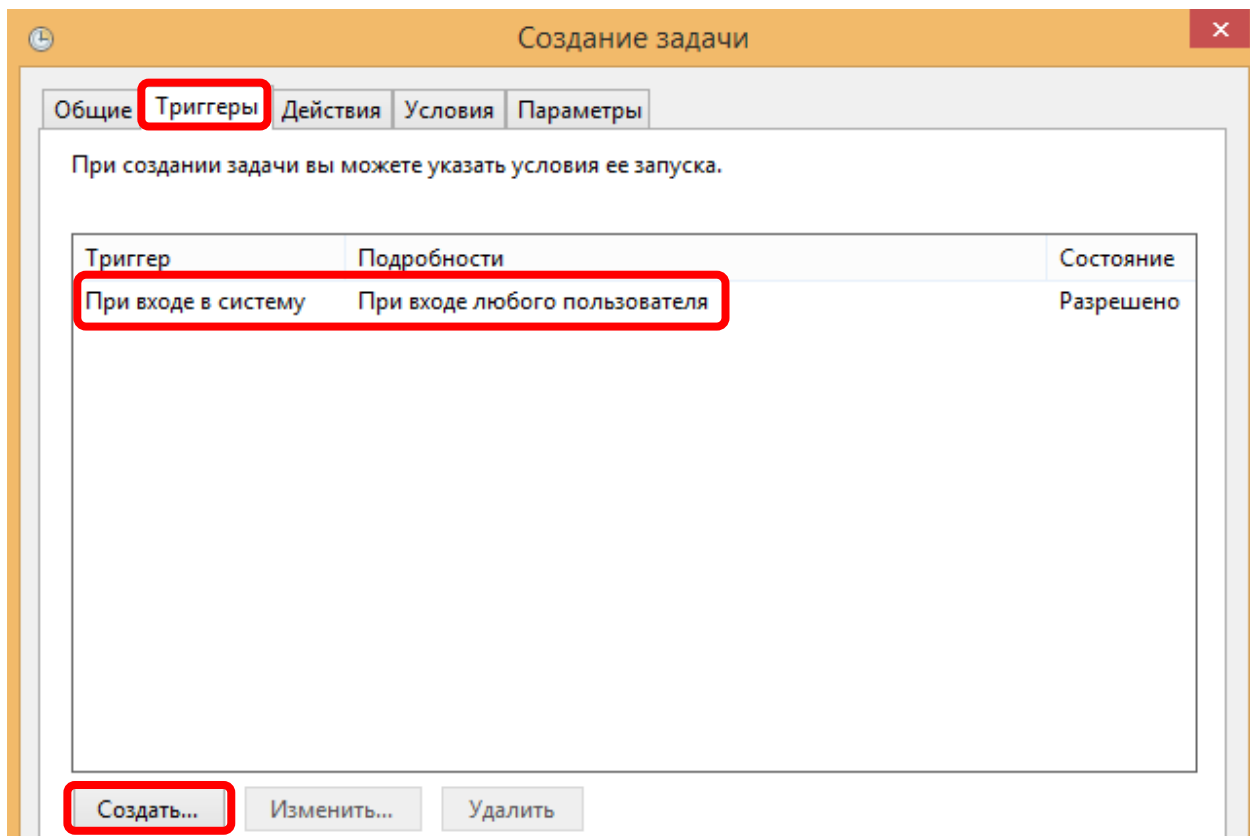


Рисунок 282 – Настройка условий запуска задачи

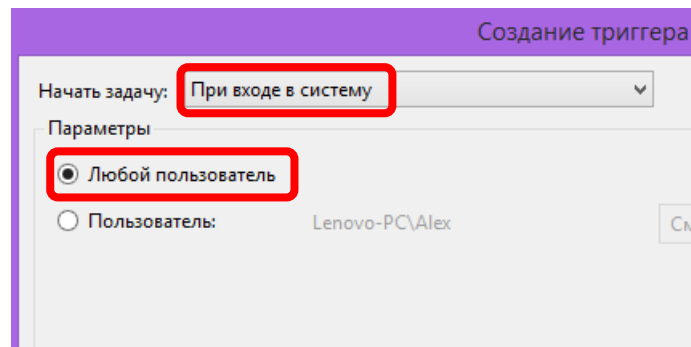


Рисунок 283 – Настройка запуска при входе в систему

На вкладке «Действия» (рис. 284) указываем, какую программу требуется запустить. Для этого необходимо нажать кнопку «Создать» и в появившемся окне (рис. 285) указать «Запуск программы» и, нажав кнопку «Обзор», выбрать адрес до WAMP-менеджера (при установке WAMP-сервера в папку по умолчанию, это адрес «C:\wamp64\wampmanager.exe»).

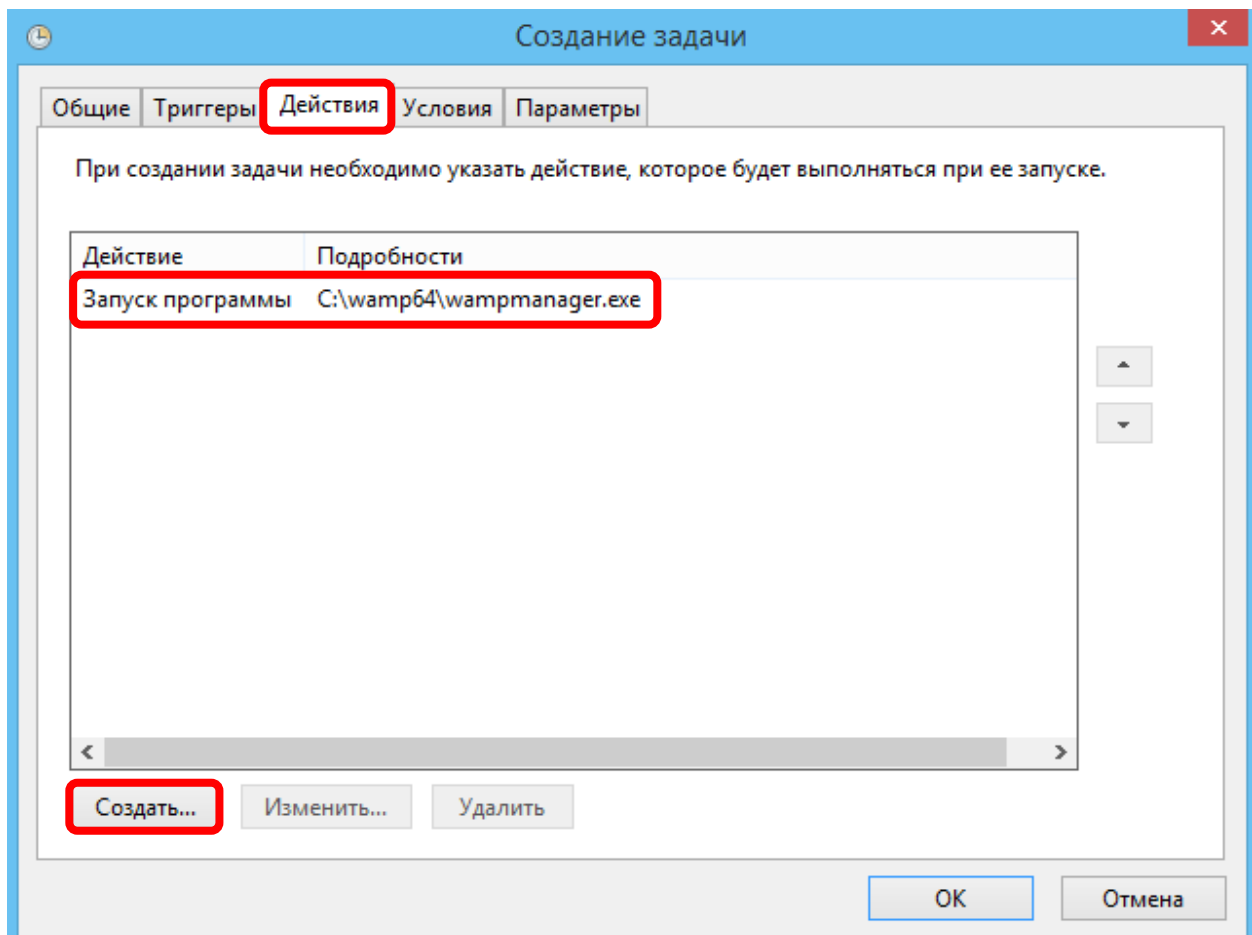


Рисунок 284 – Настройка действия, выполняемого задачей

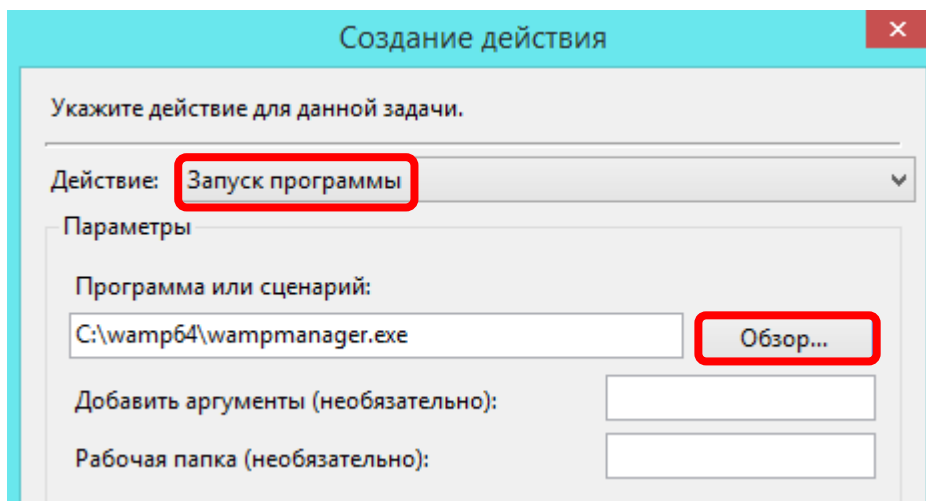


Рисунок 285 – Настройка запускаемой программы

На вкладках «Условия» (рис. 286) и «Параметры» (рис. 287) необходимо снять галочки, как показано на соответствующих рисунках.

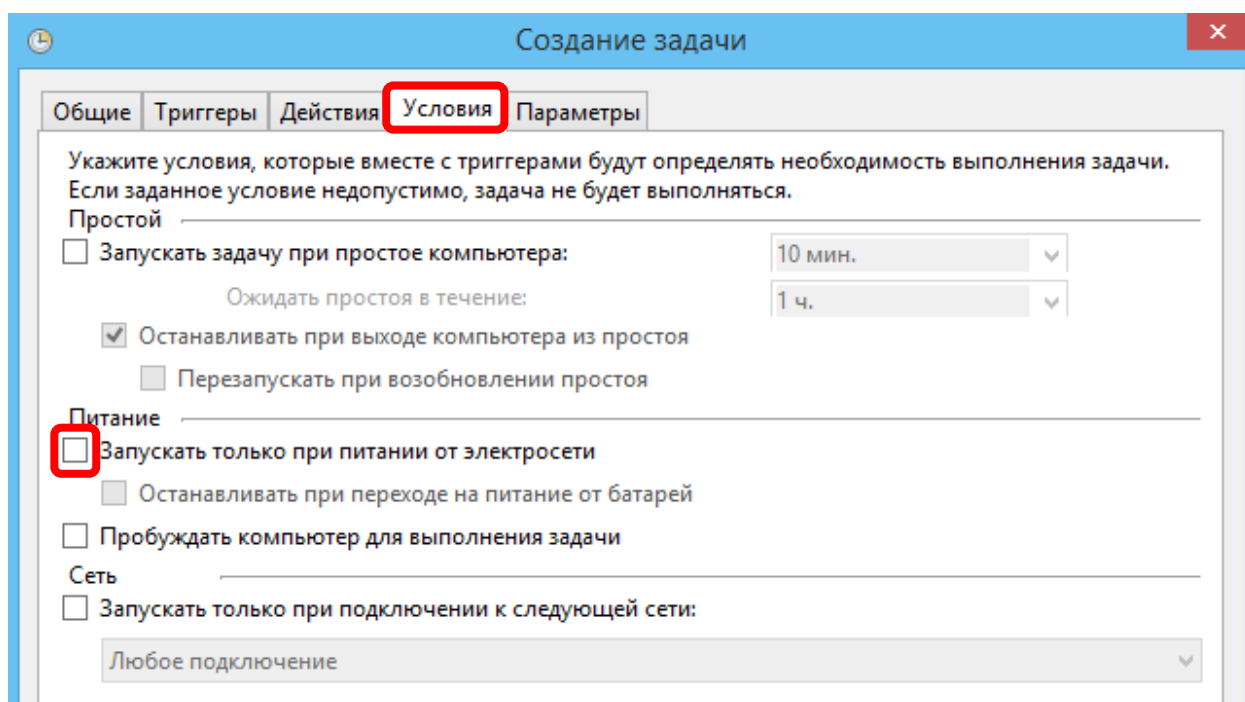


Рисунок 286 – Настройка дополнительных условий

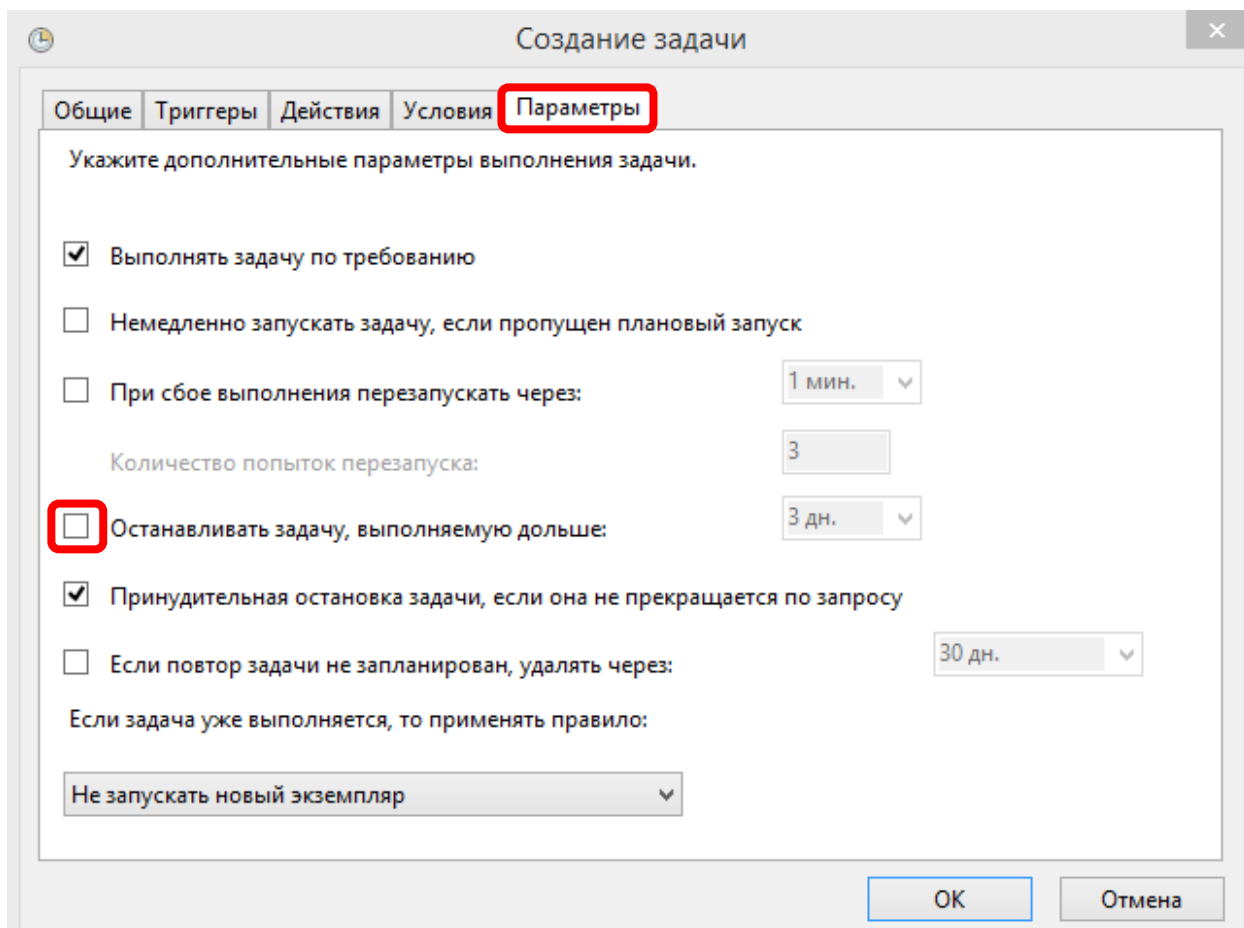


Рисунок 287 – Настройка дополнительных параметров

После нажатия кнопки «ОК», появится окно (рис. 288), в которое нужно ввести пароль администратора. **На учетной записи администратора обязательно должен быть установлен пароль! (с пустым паролем работать не будет!!!).**

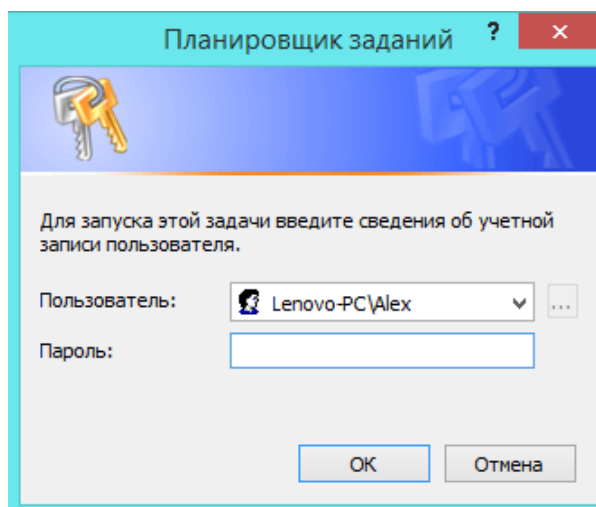


Рисунок 288 – Требование ввода пароля администратора

4. Проверка работы MySQL-сервера из учетной записи студента

После этого нужно **перезагрузить компьютер** и, войдя из учетной записи студента, проверить работу WAMP-сервера (как было описано ранее, за исключением того, что теперь не нужно запускать ярлык «**Wampserver64**» на рабочем столе).

Для проверки работы сервера необходимо открыть браузер и ввести адрес **localhost**. Далее выбрать «**PhpMyAdmin**» (рис. 277).

!!! Сервер стартует не мгновенно, после входа в учетную запись необходимо подождать загрузки сервера не менее 1 минуты.

Далее ввести имя пользователя **root** (без пароля) и подключиться к серверу **MySQL** (рис. 278).

5. Изменение и удаление созданной задачи

Для просмотра и изменения (редактирования, отключения, удаления) созданной задачи необходимо зайти (из учетной записи администратора) в Планировщик заданий и выбрать «Библиотека планировщика заданий» (рис. 289), после чего в списке (по алфавиту) найти созданную ранее задачу.

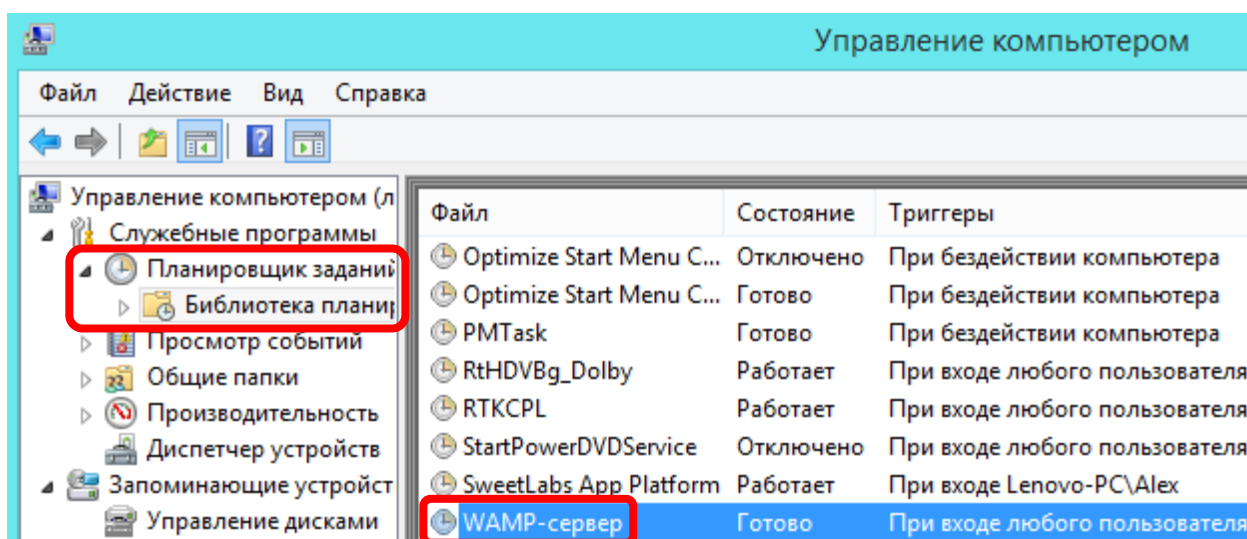


Рисунок 289 – Просмотр созданной задачи в Планировщике заданий

КОНТРОЛЬНЫЕ ВОПРОСЫ

- а) Поясните функцию перечисленных ключевых слов SQL, приведите пример использования (один пример может пояснять работу сразу нескольких ключевых слов).
- 1) CREATE;
 - 2) VALUES;
 - 3) AS;
 - 4) ASC;
 - 5) SELECT;
 - 6) DROP;
 - 7) ORDER BY;
 - 8) INSERT;
 - 9) INNER;
 - 10) PRIMARY KEY;
 - 11) LIKE;
 - 12) JOIN;
 - 13) DELETE;
 - 14) VARCHAR;
 - 15) UPDATE;
 - 16) NOT NULL;
 - 17) SET;
 - 18) OUTER.
- б) Какие из перечисленных выше ключевых слов являются первым словом в запросе? Какие являются первым словом в строке?
- в) Расшифруйте и поясните каждый из перечисленных терминов. Если термины имеют сходное значение или написание, объясните их отличия.
- 1) СУБД;
 - 2) БД;
 - 3) Access;
 - 4) Apache;
 - 5) SQL;
 - 6) MySQL;
 - 7) WAMP;
 - 8) DB;
 - 9) MariaDB;
 - 10) HTML;
 - 11) PHP;
 - 12) PhpMyAdmin;
 - 13) HTTP-клиент;
 - 14) HTTP-сервер;
 - 15) Web-сервер;
 - 16) WAMP-сервер;

- 17) SQL-сервер;
- 18) Apache-сервер.

!!! Приведенный список терминов является **минимальным** обязательным для студента. Необходимо знать все из перечисленных терминов!
Для смягчения условий оценки студентов преподаватель может разрешить студенту допустить одну ошибку, но не более.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. SQL INNER JOIN Keyword // W3Schools Online Web Tutorials : [сайт]. – 2025. – URL: https://www.w3schools.com/sql/sql_join_inner.asp (дата обращения: 05.09.2023). – Текст электронный.
2. SQL LEFT JOIN Keyword // W3Schools Online Web Tutorials : [сайт]. – 2025. – URL: https://www.w3schools.com/sql/sql_join_left.asp (дата обращения: 05.09.2023). – Текст электронный.
3. SQL RIGHT JOIN Keyword // W3Schools Online Web Tutorials : [сайт]. – 2025. – URL: https://www.w3schools.com/sql/sql_join_right.asp (дата обращения: 05.09.2023). – Текст электронный.
4. SQL Server COALESCE() Function // W3Schools Online Web Tutorials : [сайт]. – 2025. – URL: https://www.w3schools.com/sql/func_sqlserver_coalesce.asp (дата обращения: 05.09.2023). – Текст электронный.
5. CommonHTTPStatusCodes : [сайт]. – URL: <https://cwiki.apache.org/confluence/plugins/servlet/mobile#content/view/115522260> (дата обращения: 15.09.2024). – Текст электронный.

Учебное издание

Новиков Александр Игоревич

**Алгоритмизация и программирование
SQL, Apache, PHP, HTML
Часть 2**

Учебно-методическое пособие

Редактор и корректор Д. А. Романова
Техн. редактор Д. А. Романова

Темплан 2024 г., поз. 5071

Подписано к печати 25.03.2025.	Формат 60x84/16.	Бумага тип № 1.
Печать офсетная.	Печ.л. 7,1.	Уч.-изд. л. 7,1.
Тираж 30 экз.	Изд. № 18	Цена «С». Заказ №

Ризограф Высшей школы технологии и энергетики СПбГУПТД,
198095, Санкт-Петербург, ул. Ивана Черных, 4.