АКАДЕМИЯ АЙТИ

День 26

academyit.ru

Работа с файлами и каталогами. Основные операции с путями к файлам. Импорт пакета. Важнейшие стандартные пакеты. Подсистема pip . Установка стороннего модуля. Создание собственных модулей.

Турашова Анна Николаевна
Преподаватель
anna1turashova@gmail.com
Telegram: @anna1tur

Проверка домашнего задания

Задача 1.

Документ «article.txt» содержит следующий текст:

Вечерело
Жужжали мухи
Светил фонарик
Кипела вода в чайнике
Венера зажглась на небе
Деревья шумели
Тучи разошлись
Листва зеленела

Требуется реализовать функцию longest_words(file), которая выводит слово, имеющее максимальную длину (или список слов, если таковых несколько).

Задача 2.

Требуется создать csv-файл «rows_300.csv» со следующими столбцами:

– № - номер по порядку (от 1 до 300);
– Секунда – текущая секунда на вашем ПК;
– Микросекунда – текущая миллисекунда на часах.

На каждой итерации цикла искусственно приостанавливайте скрипт на 0,01 секунды.

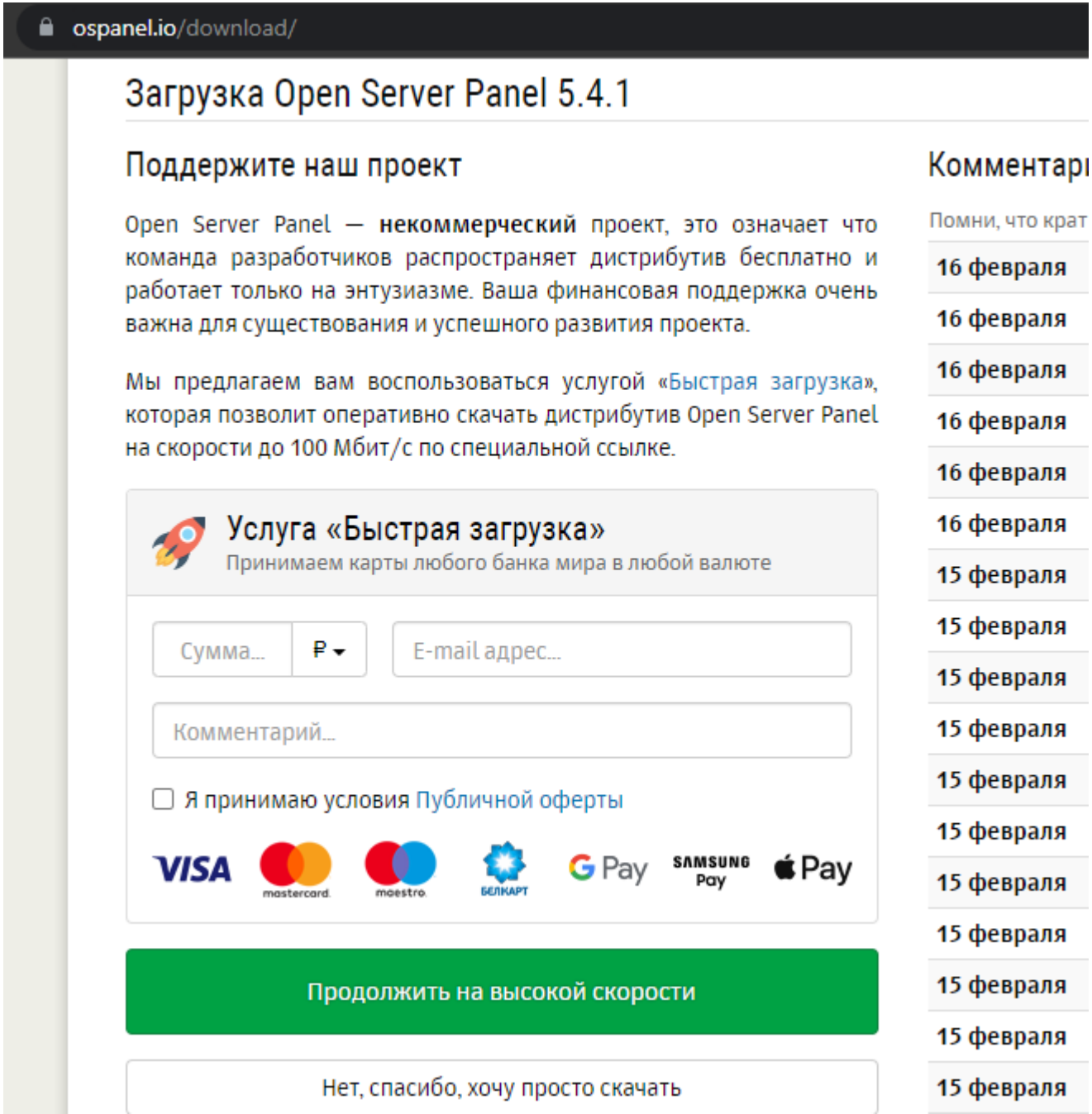
Задача 3.

Создайте файл json с любыми json данными.
Сохраните его в scv и xlsx.

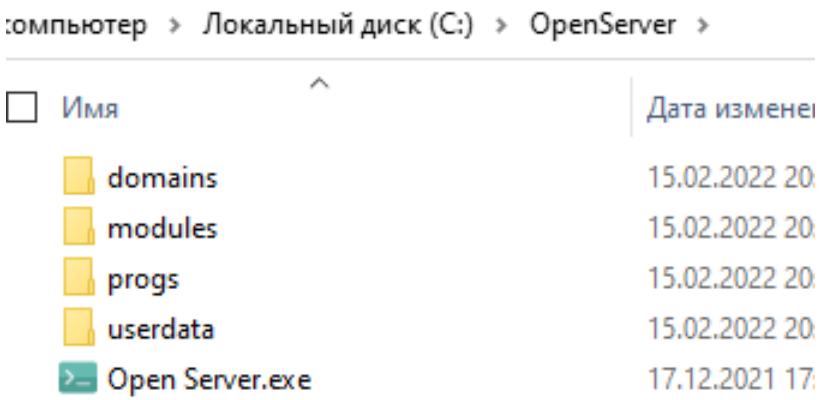
Создайте (или загрузите из файла) объект DataFrame. Попробуйте сохранить его в json.

OpenServer, phpMyAdmin

Скачиваем Open Server из любого источника. Официальный сайт:

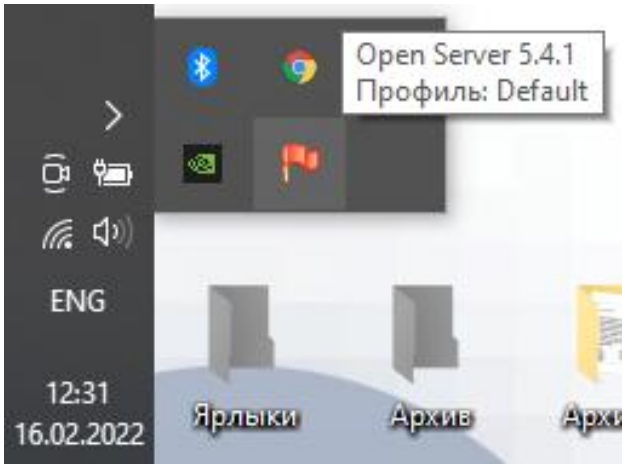


После установки нажимаем на Open Server.exe:

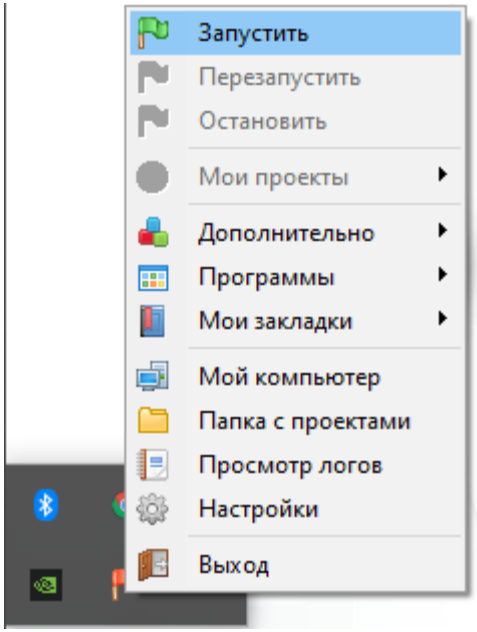


(после установки перезагрузите компьютер!)

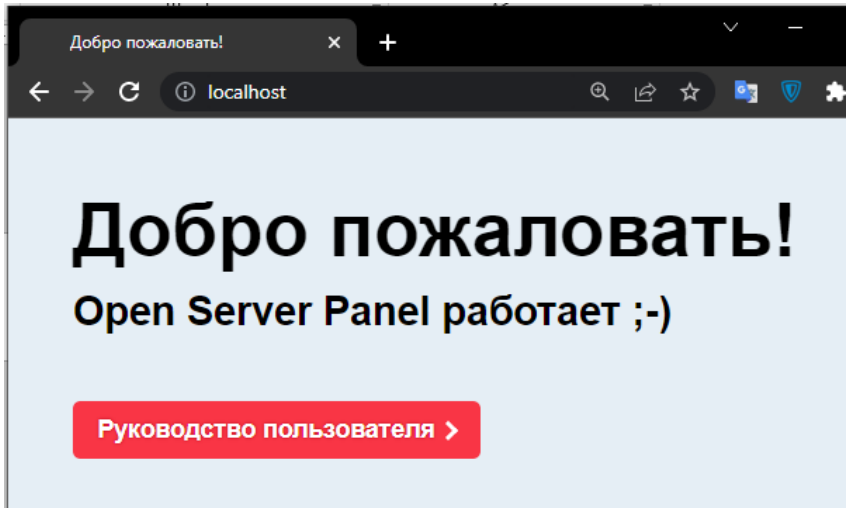
Около даты компьютера появится красный флаг – через него будем взаимодействовать с Open Server:



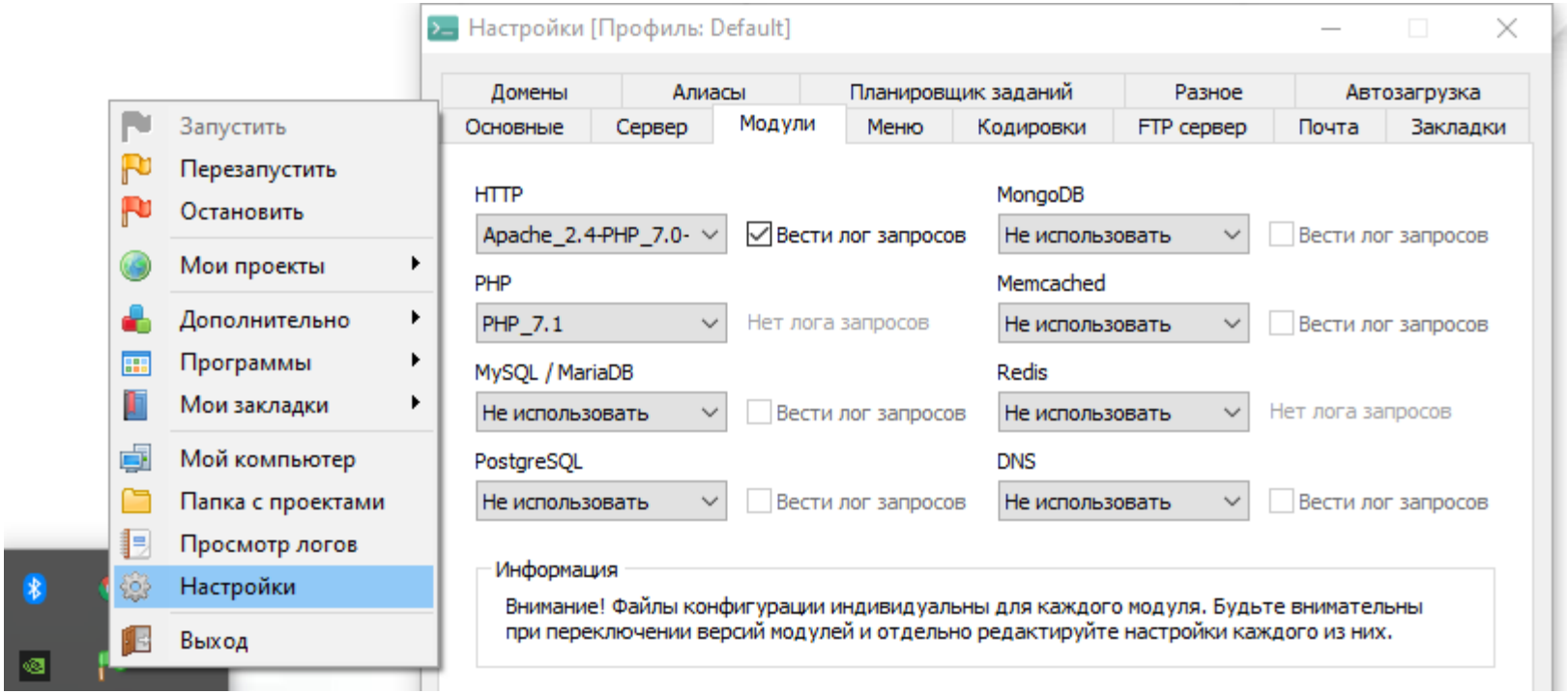
Нажмите на красный флажок и выберите пункт «запустить»



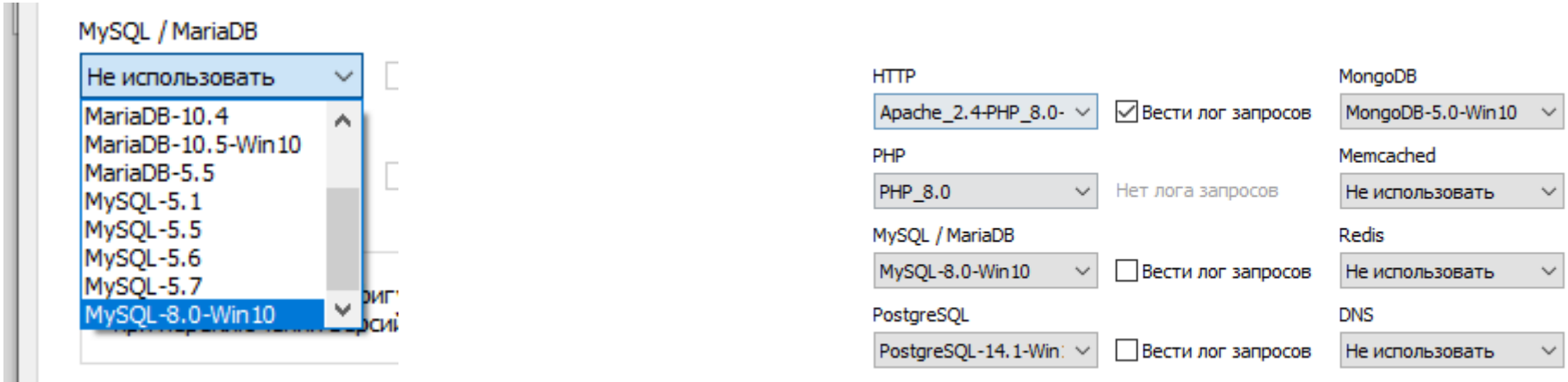
Если флажок стал зелёным, значит всё запустилось корректно.
Зайдите в браузер и напишите http://localhost/ :



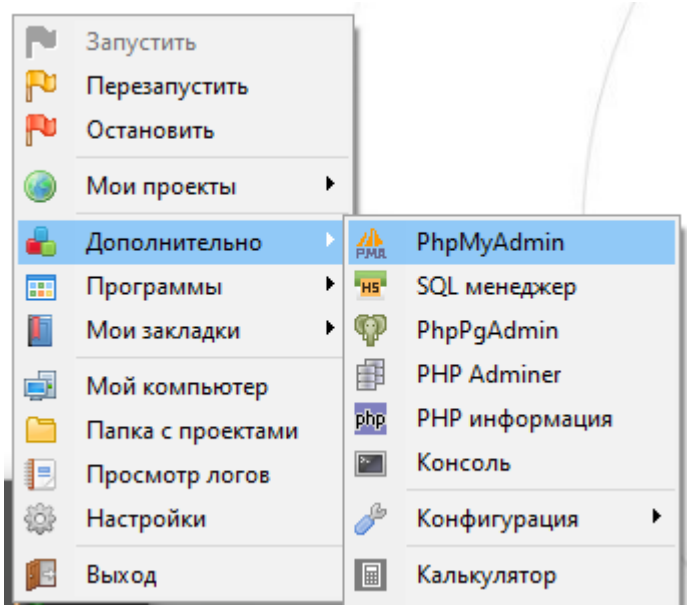
Настроем Open Server. Перейдём в настройки:

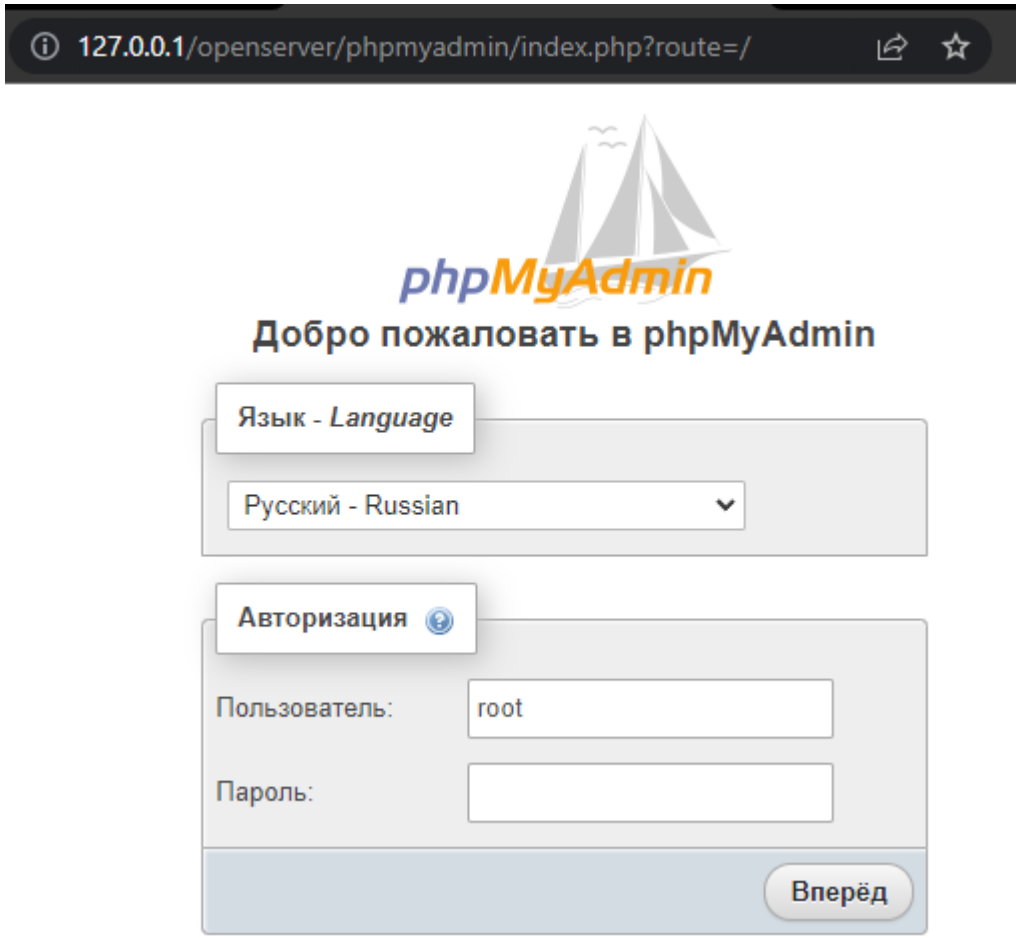


В модулях под пунктом MySQL/MariaDB ничего не выбрано. Выберите последнюю версию MySQL и нажмите «сохранить».



Теперь в «дополнительно» у вас появится PhpMyAdmin, можете нажать на него или в браузере ввести адрес http://127.0.0.1/openserver/phpmyadmin/index.php

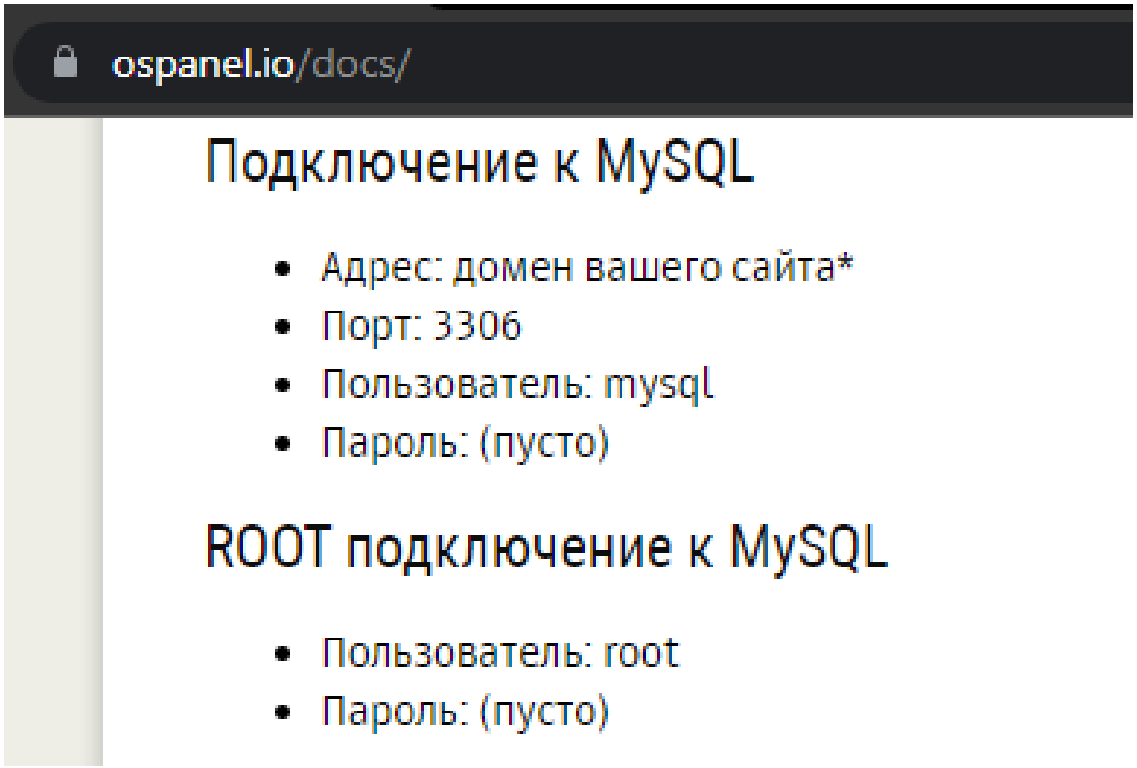




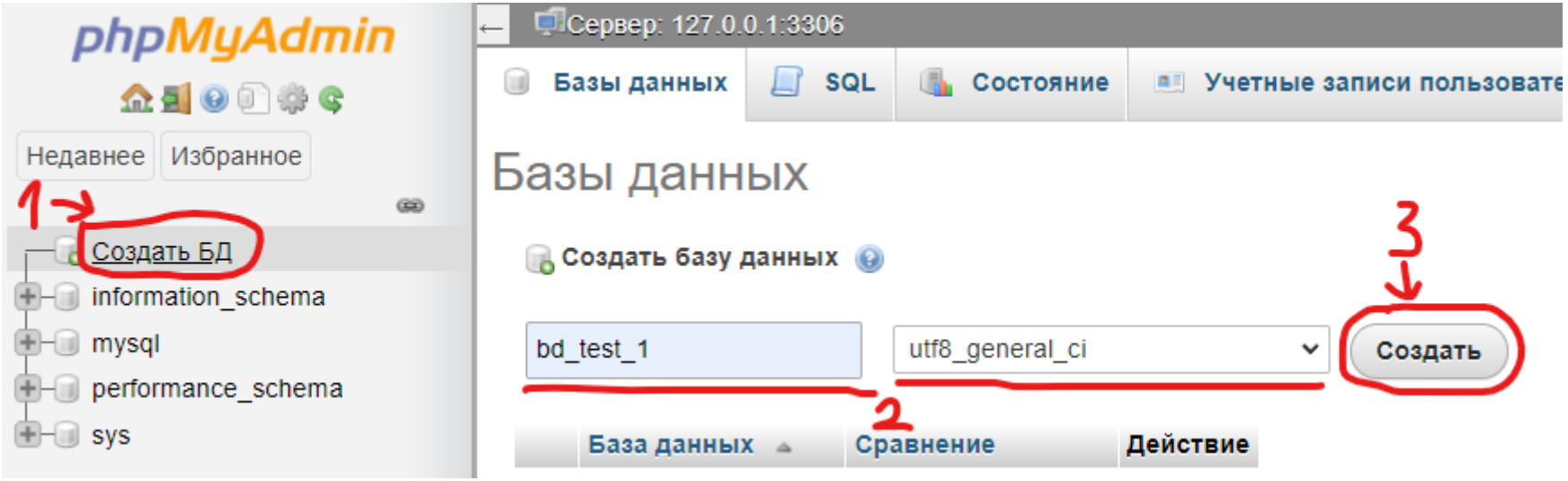
Для входа вводим данные:

Логин: root
Пароль: пусто

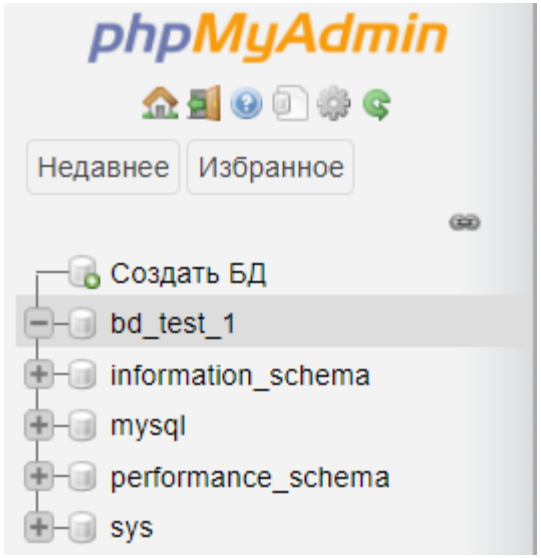
Дополнительную информацию по портам, логинам и паролям от разных баз данных, поддерживаемых Open Server, можете найти на сайте https://ospanel.io/docs/ :

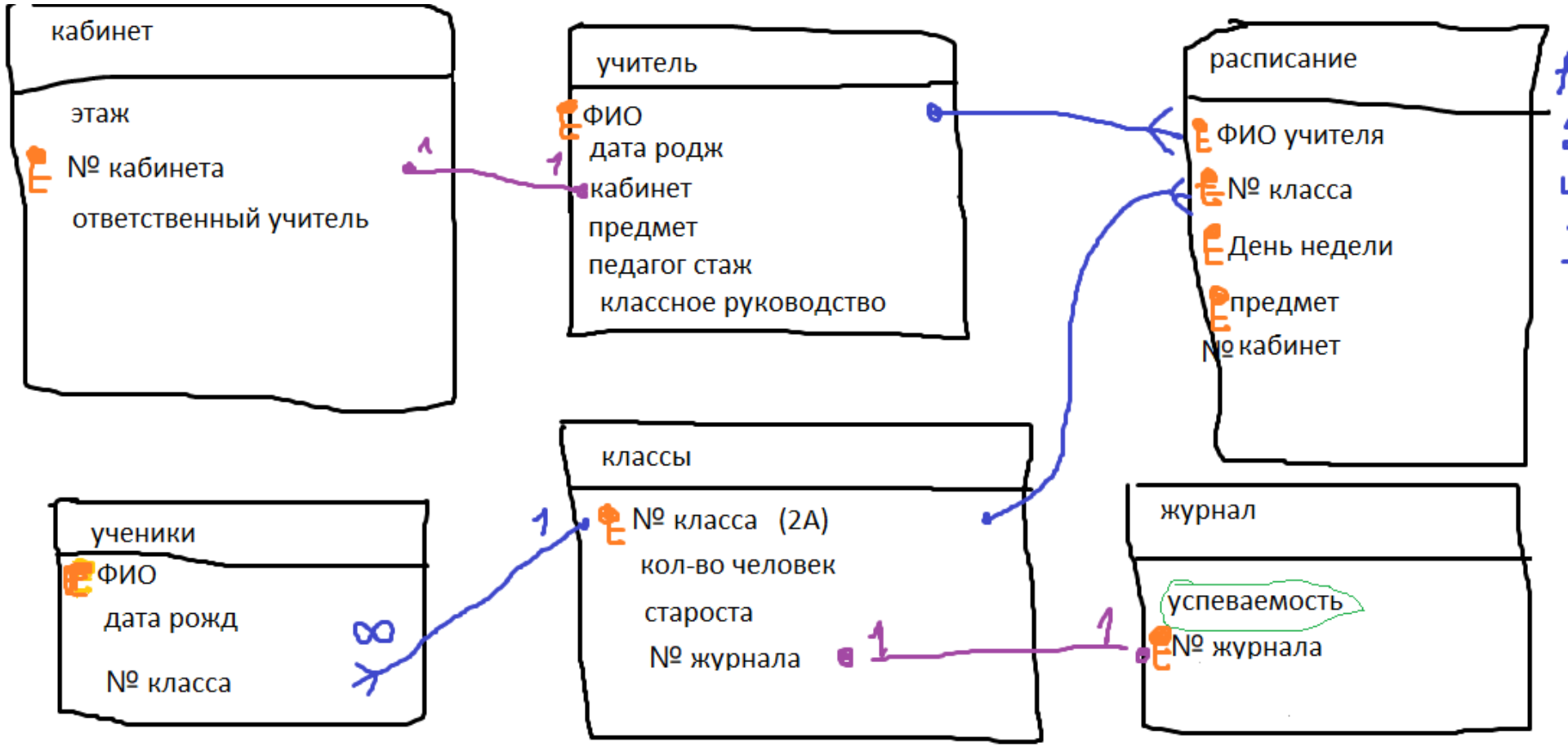


Теперь создадим свою базу данных. Нажмите «Создать БД», введите имя и выберите тип «utf8_general_ci». Нажимаем «Создать».



Теперь ваша БД в списке:




кабинет
этаж
№ кабинета
ответственный учитель
учитель
ФИО
дата родж
кабинет
предмет
педагог стаж
классное руководство
расписание
ФИО учителя
№ класса
День недели
предмет
№ кабинет
ученики
ФИО
дата рожд
№ класса
классы
№ класса (2А)
кол-во человек
староста
№ журнала
журнал
успеваемость
№ журнала
1
1
1
1
1
∞

SQL

SQL:

SQL (*Structured Query Language*) — это язык структурированных запросов. Он позволяет читать, записывать, удалять, сортировать и фильтровать информацию в базе данных.

В **SQL** используется немного слов. Он напоминает человеческий язык и поэтому его легко изучить. С его помощью можно работать с реляционными базами данных: пользователь отправляет **SQL-запрос** к базе данных через систему управления базами данных (*СУБД*). Последняя обрабатывает запрос и отправляет полученные данные пользователю.

Структура **SQL-запроса** Запрос на выборку данных выглядит вот так:

```
SELECT ('столбцы через запятую или символ * для выбора всех столбцов')
FROM ('таблицы через запятую')
WHERE ('условие или фильтр')
GROUP BY ('столбцы через запятую, по которым нужно сгруппировать данные')
HAVING ('условие в уже сгруппированных данных')
ORDER BY ('столбцы через запятую, по которым нужно отсортировать вывод')
```

SQL:

Рассмотрим подробнее, как производится выборка. (изменено)

SELECT и **FROM SELECT** и **FROM** — обязательные ключевые слова в этом запросе. С их помощью можно указать, откуда и какие данные можно выбрать: К примеру, выбрать имя из таблицы *Users*:

```
SELECT name FROM Users
```

Получить только имя и пароль зарплаты из этой же таблицы:

```
SELECT name, password FROM Users
```

Обратите внимание: имена столбцов указываются через запятую.

Выбрать все столбцы из таблицы Uses:

```
SELECT * FROM Users
```

Для выборки всех столбцов применяется групповой символ «*». При его использовании столбцы будут возвращены, но иногда порядок может не соблюдаться.

Групповой символ упрощает запрос, но при этом снижает производительность. Поэтому лучше использовать его в редких случаях.

SQL:

WHERE

Обычно нам нужна определенная информация из таблицы. Но как ее быстро найти? **WHERE** помогает извлечь информацию, отфильтровав ее по одному или нескольким условиям. Это очень удобно!

С **WHERE** применяются такие операции:

"=" (равенство);
"<>" или "!=" (неравенство);
"<" (меньше);
"<=" (меньше или равно) или !> (не больше);
">" (больше);
">=" (больше или равно) или !< (не меньше);
"BETWEEN" (между двумя значениями);
"IS NULL" (пустое поле).

Некоторые из операций приведены в нескольких вариантах, потому что в разных СУБД они указываются по-разному. Чтобы узнать, какие операции используются в вашей СУБД — смотрите ее документацию.

SQL:

Теперь вернемся к практике. Например, вам нужно выбрать имена пользователей с рейтингом свыше 1000. Применим **WHERE**:

```
SELECT name FROM User
WHERE rate > 1000
```

Если требуется указать значение строки, заключите его в апострофы:

```
SELECT * FROM User
WHERE role = ‘Guest'
```

Фильтр по нескольким условиям

Данные можно фильтровать не только по одному, а и по нескольким условиям и значениям. Для этого используются операторы IN, NOT IN, AND, OR.

Отфильтровать по нескольким значениям с дополнительными условиями:

```
SELECT * FROM User
WHERE role IN (‘Admin’, Moderator')
```

Отфильтровать по нескольким значениям с исключением:

```
SELECT * FROM User
WHERE role NOT  IN (‘Admin’, Moderator')
```

Выбрать пользователей с ролью гостя и с рейтингом свыше 1000:

```
SELECT * FROM Users WHERE role = ‘Guest' AND rate > 1000
```

SQL:

GROUP BY С помощью необязательного предложения **GROUP BY** создаются группы данных. Это удобно для получения итоговых значений. Например, нужно узнать, сколько человек зарегестрированно. Инструкция может выглядеть так:

```
SELECT role, COUNT (*) AS cnt
FROM Users
GROUP BY role
```

Этот код возвращает названия ролей и количество пользователей с каждой из них. Количество пользователей помещается в столбец с псевдонимом **cnt**, который мы задали с помощью ключевого слова **AS**.

Предложение **GROUP BY** указывается после **WHERE** и перед **ORDER BY**.

В **GROUP BY** можно указать столько столбцов, сколько нужно. В результате группы вкладываются друг в друга.

При вложении данные будут суммироваться для последней заданной группы, а не для отдельно для каждого столбца.

В предложении **GROUP BY** можно указать только столбцы выборки или выражения. В нем не указывается функция группирования и не применяются псевдонимы.

Если в столбце, по которому производится группирование, встречается одна или несколько строк со значением **NULL**, они выделяются в отдельную группу.

SQL:

HAVING

С помощью предложения **GROUP BY** можно также указывать, какие группы включить в результат, а какие — исключить из него. Для этого используется предложение **HAVING**.

Оно очень напоминает **WHERE**, но фильтрует не строки, а группы.

HAVING можно использовать с любыми операторами. В этом предложении используется тот же синтаксис, что и в предложении **WHERE**:

```
SELECT role, COUNT (*) AS cnt
FROM Users
GROUP BY role
HAVING COUNT(*) >= 3
```

Этот код похож на предыдущий, но возвращает только те группы, в которых найдены три или больше сотрудников. Фильтрация выполняется по итоговому значению группы. Этим **HAVING** отличается от **WHERE**, которое фильтрует по значениям строк.

Эти предложения можно использовать вместе. Например, можно узнать, сколько пользователей с ролью в которой более трех человек, с рейтингом более 1000:

```
SELECT role, COUNT (*) AS cnt
FROM Users
WHERE rate > 1000
GROUP BY role
HAVING COUNT(*) >= 3
```

SQL:

ORDER BY

Предложение **ORDER BY** используется для сортировки результатов запроса. В нем указываются имена столбцов, по которым нужна сортировка.

Давайте отсортируем список имён пользователей:

```
SELECT name FROM Users
ORDER BY name
```

В предложении ORDER BY можно указывать и те столбцы, которые не выбраны в операторе **SELECT**:

```
SELECT name FROM Users
ORDER BY rate
```

Так список имён будет отсортирован по рейтингу.
Сортировку можно выполнять и по нескольким столбцам. Для этого имена столбцов указывают через запятую:

```
SELECT name FROM Users
ORDER BY rate, name
```

Так мы увидим список пользователей, который сначала отсортирован по рейтингу, а затем — по имени.

SQL:

Вместо имен столбцов можно указать их порядковые номера в операторе **SELECT**:

```
SELECT name FROM Users
ORDER BY 2, 1
```

Сортировка по убыванию

В предыдущих примерах мы сортировали по возрастанию (это делается по умолчанию). Но можно сортировать и по убыванию. Для этого укажем слово **DESC**:

```
SELECT name FROM Users
ORDER BY rate DESC
```

Если обратная сортировка выполняется по нескольким столбцам, укажите ключевое слово **DESC** после каждого из них.

Слово **DESC** — это сокращение от слова **DESCENDING**. В запросах можно использовать как полную, так и сокращенную форму. Для сортировки в порядке возрастания тоже существует ключевое слово. Его полная форма — **ASCENDING**, а сокращенная — **ASC**. Поскольку по умолчанию выполняется сортировка по возрастанию, то это слово не указывают.

SQL:

Объединение таблиц

Иногда нам нужны данные из нескольких таблиц. Рассмотрим пример:

```
SELECT name, rate, post_id
FROM Users, Posts
WHERE Users.user_id = Posts.user_id
```

Этот код возвратит имена пользователей из таблицы Users и номера постов из таблицы Posts, которые выполнены соответствующими пользователями. В предложении **WHERE** имена столбцов указаны с именами соответствующих таблиц. Это необходимо, чтобы СУБД могла различать столбцы user_id из разных таблиц. Такое объединение называется внутренним. Для него можно использовать специальный синтаксис с ключевым словом **INNER JOIN**. Приведенный ниже код выдаст те же результаты, что и предыдущий фрагмент:

```
SELECT name, rate, post_id
FROM Users
INNER JOIN Posts ON Users.user_id = Posts.user_id
```

Вместо предложения **WHERE** используется предложение **ON**, синтаксис которого совпадает с синтаксисом **WHERE**.

Число объединяемых таблиц в **SQL** не ограничено, но может ограничиваться в разных СУБД. Обратите внимание: чем больше таблиц объединяется, тем ниже производительность. Поэтому не рекомендуем объединять таблицы без особой необходимости.

SQL:

SQL — простой для освоения и при этом мощный язык.

Он появился в 1970-х и до сих пор используется, хотя наряду с ним появляются новые похожие языки.

Этот язык используется различными *СУБД*: ***MySQL***, ***SQLite***, ***PostgreSQL***.

АПЛАНА
ГРУППА КОМПАНИЙ

Входит в ГК Аплана

АКАДЕМИЯ АЙТИ

Основана в 1995 г.

E-learning и очное обучение

Направления обучения:

- Информационные технологии
- Информационная безопасность
- ИТ-менеджмент и управление проектами
- Разработка и тестирование ПО
- Гос. и муниципальное управление

Филиалы:

Санкт-Петербург, Казань, Уфа, Челябинск, Хабаровск, Красноярск, Тюмень, Нижний Новгород, Краснодар, Волгоград, Ростов-на-Дону

Головной офис в Москве

Разработка программного обеспечения и информационных систем

Ежегодные награды Microsoft, Huawei, Cisco и другие

Программы по импортозамещению

Ресурсы более 400 высококлассных экспертов и преподавателей

Сеть региональных учебных центров по всей России

Крупные заказчики

СБЕР

РЖД

Билайн

TATNEFT

НОРНИКЕЛЬ

100+
сотрудников

АКАДЕМИЯ АЙТИ

Спасибо за внимание!

Центральный офис:
Москва, Варшавское шоссе 47, корп. 4, 7 этаж
Тел: +7 (495) 150-96-00
academy@it.ru
academyit.ru